TECHNICAL SUMMARY

digital

DECsystem 10

The Contents

1
Introduction

The DECsystem-10 Technical Summary introduces the characteristic features and capabilities of the DECsystem-10 to computer analysts and system programmers. Application programmers, system managers, and system operators may also use this summary as a tool to become familiar with the components, services, and operations of the DECsystem-10.

This technical summary is a detailed introduction to all aspects of the DECsystem-10 system — from the DECsystem-10 processors and peripherals devices to the TOPS–10 operating system and DIGITAL's support services. The technical summary is primarily intended for the system programmer and computer system specialists who are already familiar with computer hardware and software. However, it contains useful information for application programmers, system managers, and system operators.

You are encouraged to read this technical summary selectively. Many of the system's concepts and features are repeated throughout the text in different contexts. You might first skim through the summaries to find those topics that interest you most, perhaps by reading just the abstracts that appear at the beginning of each section. You can then start with the sections of interest, knowing which section to refer to when you come across references to concepts discussed elsewhere.

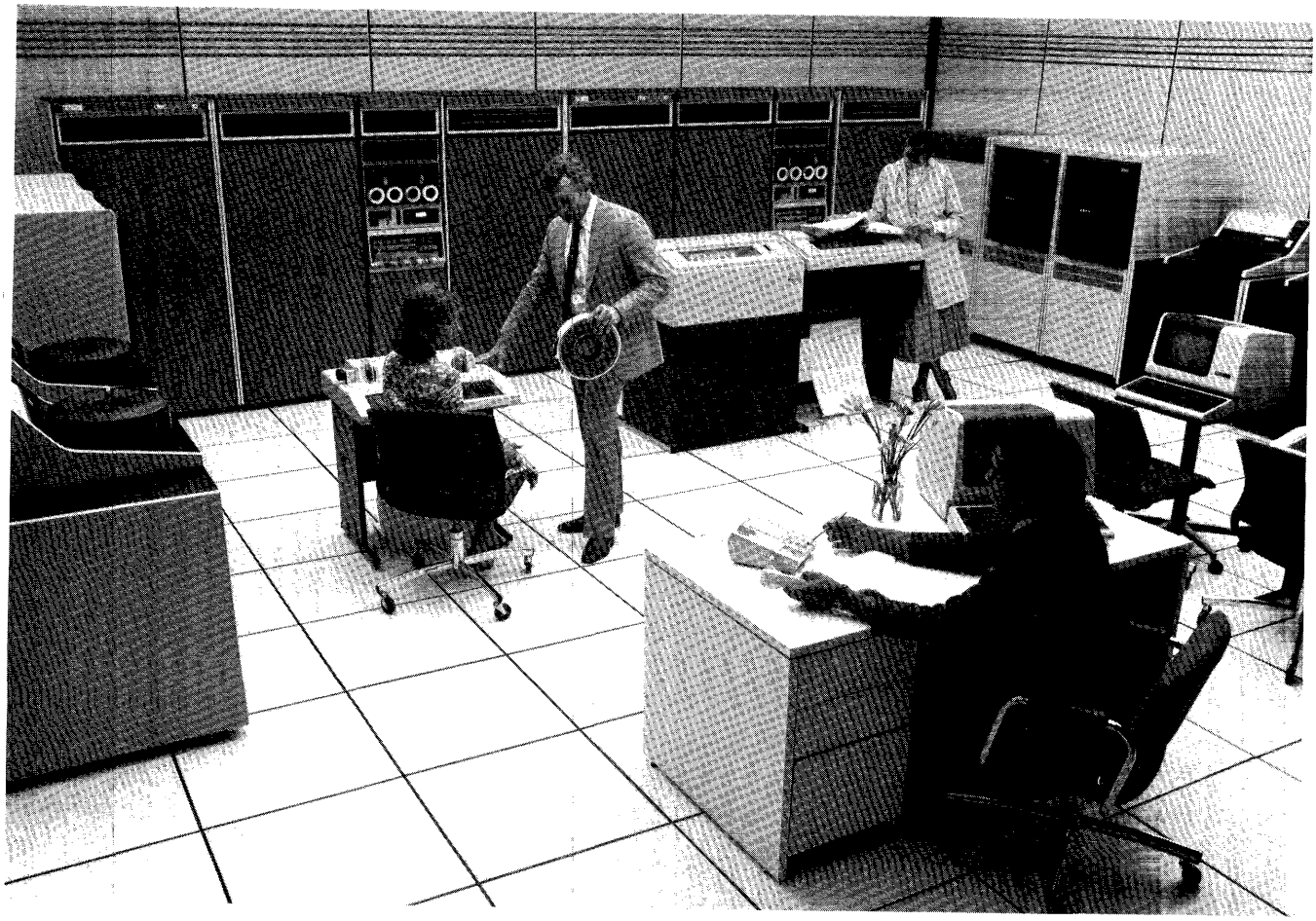
If you are familiar with computer industry terminology and simply want an overview of the DECsystem-10 features, you should read THE SYSTEM section. This section briefly describes the DECsystem-10 characteristics and introduces features of the system that are described in detail throughout the remainder of the technical summary.

Some people may find it more helpful to begin with THE USERS section. This section introduces many of the aspects of the system that support application programming, application tools, system progamming, system management, and operator control.

If you are an application programmer, you will find the sections on high-level languages, data management, and application products provide an in-depth discussion on the system's characteristics and capabilities.

Finally, if you are considering obtaining a DECsystem-10 or have a system now, you should read the SUPPORT SERVICES section to become familiar with the kinds of services DIGITAL makes available to its computer system users.

2
System Overview



The DECsystem-10 is a multipurpose system. It provides a wide range of computing versatility and power combined with exceptional reliability and efficiency. The system has built-in protection mechanisms in both the hardware and software to ensure data integrity and system availability. On-line diagnostics and error detection and logging verify system integrity. Many hardware and software features provide rapid diagnosis and automatic recovery should the power, hardware, or software fail.

The central processor is based on a 36-bit architecture. The processor's instruction set is implemented in microcode. All instructions are capable of directly addressing 256K words of memory without resorting to base registers, displacement addressing, or indirect addressing. The instruction set includes a large number of operation codes. These codes designate the operation to be performed and allow for a full range of operating system functions.

The operating system that runs on the DECsystem-10 is called TOPS–10. TOPS–10 is easy to use, and provides a highly reliable virtual memory, multipurpose operating system. TOPS–10 contains the features to support full-language timesharing for program development and various types of interactive and terminal-oriented applications, plus a full capability, multiprogram batch system.

TOPS–10 is both flexible and extendable. Virtual memory features enable the programmer to write large programs that can execute in both small and large memory configurations without requiring the programmer to define overlays or later modify the program to take advantage of the additional memory.

The DECsystem-10 is a state-of-the-art hardware and software system. It serves interactive timesharing users and performs multiprogramming batch processing simultaneously. Because of the DECsystem-10 versatility, it can be used for a variety of applications, such as:

- Commercial applications, which require decimal arithmetic, variable length fields, and editing capabilities along with data integrity, system operation, and high throughput
- Scientific applications, which require larger storage capacity and high-speed computational capabilities
- Communications, which require a large number of communication terminals and fast response

The design of the DECsystem-10 is open-ended. This design permits the DECsystem-10 to be easily expanded to incorporate new features, devices, and technology.

The DECsystem-10 offers system compatability. This characteristic permits programs that run on one version of the TOPS–10 operating system to operate on subsequent versions of the operating system without any program changes.

The DECsystem-10 allows for a variety of I/O devices to be connected to the system. The hardware features integrated high-speed data channels and mass storage controllers for disk drives and magnetic tapes.

COMPONENTS

The following are the components that make up the DECsystem-10:

- Processors — includes two models of CPUs that run the TOPS–10 operating system; the KS10 and the KL10. Both processors use the same instruction set and support MOS memory.
- Peripherals — includes a wide variety of small and large capacity disk drives, magnetic tape systems, hard copy and video terminals, line printers, card readers, a paper-tape-punch/reader, and a plotter.
- Operating System — includes a virtual memory manager, command interpreter, scheduler, swapper, monitor call handler, file handler, system services, device drivers, and operator's and system manager's tools.
- Languages — includes DECsystem-10 MACRO assembly language and optionally FORTRAN–10, COBOL–68 and –74, ALGOL–10, APL–10, CPL, BLISS–36, and BASIC. Development tools for programs include editors, linkers, and debuggers.
- Data Management Tools — includes DBMS software and IQL software.
- Communications — includes DECnet-10 network software, ANF–10 software, and 2780/3780 emulation and termination.

Processors

The KL10 and KS10 CPUs form the basis of the DECsystem-10 computer systems. Both CPUs provide 36-bit addressing, eight sets of 16 general purpose registers, and seven priority interrupt levels. Both CPUs feature a microprogrammed instruction set capable of directly addressing 256K words.

The KL10 and KS10 CPUs support cache memory. The cache memory implementation for the KL10 is 2048 words. The cache memory implementation for the KS10 is 512 words. Cache memory is used to provide a faster effective memory access time.

Both the KL10 and KS10 support KLINIK, which is the remote diagnosis capability. DIGITAL Field Service Engineers can examine memory without interfering with the normal operation of the system.

The KL10 has a PDP–11 based Console/Diagnostic Front-End processor. The front-end processor plays a key role in the operation and maintenance of the KL10 processor. The front-end processor provides all console functions for the KL10 and the TOPS–10 operating system.

The KS10 has an 8-bit microprocessor for a console. The KS10 console is an extremely important subsystem because it performs all console and diagnostic functions. To allow programming of the console, an 8K Programmable Read-Only Memory (PROM) and 1K of Random Access Memory (RAM) are provided.

Two KL10 processors can be configured as a Symmetric Multi-Processor System. Symmetric Multi-Processing (SMP) provides improved system availability and performance over other types of dual-processor systems. SMP configurations may have up to 175 active jobs and 512 dedicated application terminals. With SMP, input/output devices can be connected to both CPUs; thus, if one CPU fails, the system can dynamically reconfigure to the operational CPU.

SMP systems, fully supported by the TOPS–10 operating system, offer attractive, economical solutions for increased reliability, availability, and performance. In these configurations, two processors share the workload on an almost equal basis. The processor on which the monitor is loaded is called the "boot" or "policy" CPU. This CPU has slightly more responsibility than the other CPU because it performs command decoding, swapping decisions, and a small number of other tasks. In all other respects, each CPU in a multiprocessing system is equivalent to the other. This symmetry includes processing operating system calls and performing I/O.

In an SMP configuration, it is possible for any CPU in the system, including the policy CPU, to fail without causing the system as a whole to fail. If the policy CPU fails, the other CPU notices that it has failed, and automatically assumes the responsibility of the policy CPU. If the CPU failure is corrected, the CPU can be restarted without the necessity of reloading the entire system. In addition, if the peripheral devices are appropriately dual-ported, portions of the system can be

logically and physically removed for preventive maintenance. It should be noted that multiprocessor systems offer much more than a performance increase. They also offer significant advantages in the areas of increased reliability and availability.

The Operating System

TOPS–10 allows a large number of users to be engaged in diverse applications involving many different programs and languages. The TOPS–10 operating system is designed for many applications, including computation, data, transaction, and batch processing.

TOPS–10 is designed for timesharing and concurrent batch, real-time, and remote communications in either a single or SMP configurations. TOPS–10 services interactive users, operates local and remote batch stations, and performs data acquisitions and control functions for on-line applications and other real-time projects. By dynamically adjusting system operation, TOPS–10 provides many features for each class of user and is therefore able to meet a large variety of computational requirements.

The TOPS–10 operating system allows interactive users to take maximum advantage of the timesharing environment. The system allows many independent users to share the facilities of the system simultaneously. Because of the interactive, conversational, and rapid response nature of timesharing, a wide range of tasks, from solving simple mathematical problems to implementing complete and complex information gathering and network processing, can be performed by the user. The number of users allowed on the system at any one time depends on the system configuration.

By allowing resources to be shared among users, the timesharing environment utilizes processor time and system resources that are wasted in single-user systems. Users are not restricted to a small set of system resources, but instead are provided with a full variety of facilities. By interacting with a terminal, the user has on-line access to most of the system's features. This on-line access is available through the operating system's command language, which is the means by which the timesharing user communicates with the computer system.

Through the command language, users control the running of their jobs to achieve the results they desire. Users can create, edit, and delete their files; start, suspend, and terminate their jobs; compile, execute, and debug their programs. In addition, since multiprogramming batch software accepts the same command language as the timesharing software, any user can enter a job into the batch run queue. Thus, any timesharing terminal can act as a remote job entry terminal.

Timesharing under TOPS–10 is designed in such a way that the command language, input/output processing, file structures, and job scheduling are independent of the programming language being used. In addition, standard software interfaces make it easy for the user to develop his own special language or systems. The general purpose approach is demonstrated by the many programming languages implemented by DECsystem-10 customers.

TOPS–10 provides a flexible, easy-to-use, and powerful batch processing system. The batch system includes input spoolers, batch-stream controllers, queue manager, scheduler, and output spoolers. Batch jobs can be initiated and monitored from any terminal, or they can be automatically submitted by the system at a specific time.

The TOPS–10 batch software employs many of the system's features in order to operate with maximum efficiency. Because memory is not partitioned between batch and timesharing jobs, batch jobs can occupy any available memory. Fast throughput for high-priority batch jobs is accomplished by the same swapping technique used for rapid response to interactive users. When available memory is not large enough for a high-priority batch job, TOPS–10 transfers lower priority jobs to secondary storage in order to make room for the high-priority job. Batch jobs can be designated as "background batch" so they become a lower priority than any other jobs on the system. If any other job needs memory occupied by a background batch job, TOPS–10 transfers the background batch job to secondary storage. This input/output transfer is done at the same time the processor is working on another job. Thus, processing can be overlapped to utilize time that would otherwise be wasted. Batch jobs can also share programs with timesharing and other batch jobs. Only one copy of a sharable program need be in memory to service any number of batch and timesharing jobs at the same time.

Although batch jobs are entered sequentially into the batch system, they are not necessarily run in the order that they are read because of priorities, either set by the user or computed by the system when determining the scheduling of jobs. Occasionally, the user may wish to submit jobs that must be executed in a particular order; in other words, the execution of one job is dependent on another. To ensure that jobs are executed in the proper order, the user must specify an initial dependency count for any job that is dependent upon another submission to the batch system. During the execution of the job on which the dependent job depends, the dependency count may be modified. When the dependency count equals zero, the dependent job is executed.

Programs in the batch system require little or no operator intervention. However, the operator can exercise a great deal of control if necessary. He can specify the number of system resources to be dedicated to batch processing by limiting the number of programs and both the memory and processor time for individual programs. He can stop, continue, abort, cancel, requeue, and modify its priority at any point. By examining the system queues, he can determine the status of all batch jobs.

The TOPS–10 Inter-Process Communication Facility (IPCF) provides the capability for jobs to communicate with one another. For example, if several programs are involved in processing or maintaining a data base, it is possible that one program might want to inform other programs of any modifications it made to the data. A job using the IPCF facility cannot make any changes to another job, so protection is in no way sacrificed when using this facility.

TOPS–10 allows for simultaneous operation of multiple remote stations. Software provisions are incorporated into the operating system to differentiate one remote station from another. By utilizing peripheral devices at various stations, the user is provided with increased capabilities. For example, data can be collected from various remote stations, compiled and processed at the central computer, and then the results of the processing can be sent back to the remote stations.

Remote station use of the central computer is essentially the same as local use. All sharable programs and peripherals available to local users at the central computer are also available to remote users. The remote user specifies the resources he wants to use and, if available, these resources are then allocated to the remote user in the same manner as to a local user. In addition to utilizing the peripheral devices at the central computer, the remote user can access devices located at his location or at another remote station. Local users can also make use of the peripheral devices at remote stations. Therefore, by specifying the station number in appropriate commands to TOPS–10, each user is given considerable flexibility in allocating system facilities and in directing input and output to the station of his choice.

The TOPS–10 operating system is made up of a number of separate, independent, yet interrelated modules. Some of these routines are cyclical in nature and are repeated at every system clock interrupt to ensure that every user of the computing system is receiving the requested services in a timely fashion. These cyclic routines are:

- The Command Processor
- The Scheduler
- The Swapper

The Command Processor is the communications link between the user and the operating system. Because all requests for system resources are initiated through the Command Processor, it is the most visible part of the system to the user. When the user gives commands from an interactive terminal or from a batch control file, the characters are stored in an input buffer in the operating system. The Command Processor examines these characters in the buffer, checks them for correct syntax, and invokes the resource as specified by the command.

The Scheduler is responsible for deciding which user (or users in a dual-CPU SMP configuration) job(s) should run at any given time. In addition, the Scheduler allocates shareable system resources, and saves and restores conditions needed to restart a program that has been previously suspended.

The TOPS–10 operating system employs a technique whereby jobs can exist on a secondary storage device, such as disk, as well as in memory. Therefore the scheduler decides not only what job is to be run next, but also notifies the Swapper when a job is to be swapped out onto disk.

All jobs in the system are retained in ordered groups called queues. These queues have various priorities that reflect the status of each job at any given moment. The queue in which a job is placed depends on the system resource for which it is waiting; and, because a job can wait for only one resource at a time, it can be in only one queue at a time.

The Swapper is responsible for keeping the jobs most likely to be run in memory. It determines if a job should be in memory by scanning the various queues in which a job may be. If the Swapper decides a job should be brought into memory, it may have to take another job already in memory and transfer it to secondary storage. Therefore the Swapper is not only responsible for bringing the job(s) into memory, but is responsible for selecting the job(s) to be swapped out of memory.

The noncyclic routines of the TOPS–10 operating system are invoked only by user programs or TOPS–10 monitor requirements and are responsible for providing these programs with the services available through the operating system. The noncyclic routines are:

- The UUO handler
- The Input/Output routines
- The file handler

The UUO handler (A UUO is an operating system call for I/O or other services) is the means by which a program communicates with the operating system in order to have a service performed. Because of the multitask design of TOPS–10, there are certain functions that can only be performed by the operating system to prevent user conflict and provide significant advantages in the ease of I/O programming. Communication between the user and the operating system is by way of these programmed operators (UUOs) contained in the user program which, when encountered, transfer control to the operating system for processing.

The Input/Output routines are the modules responsible for directing data transfers between peripheral devices and user programs in memory. Since all I/O is performed by the operating system (except in special real-time applications), the user is freed from the responsibility of fully understanding the detailed operation of a given device, and must only conform to a data and command protocol within the operating system.

The file handler adds permanent user storage to the computing system by allowing users to store programs and data as named files. To access these files, the user need only specify the file name and user identification. The operating system is responsible for all physical placement of these files. The user never need know where on the storage media these files are located.

These operating system modules interface and communicate to one another through a precisely defined scheme of hardware and software interrupts, and monitor traps. Figure 2–1 provides a broad overview of the interrelationships between the various modules of the TOPS–10 operating system.

Users of TOPS–10 have a wide selection of communications capabilities to enhance or facilitate their computing needs. Communications functionality has always been an integral part of the TOPS–10 operating system. With its emphasis on interactive, multimode computing, user access to the features and functionality of the operating system has always been an important design consideration of any implementation.

Within the multitask environment of TOPS–10, the following functionality exist:

- Asynchronous communication — for interactive timesharing, program development and debugging, data entry, and others
- Synchronous communication — for connection of remote batch, remote terminal concentration, and computer-to-computer links

Within the TOPS–10 environment, the operating system, not the user, handles the communications housekeeping, and all communication functions are fully supported within the operating system itself. Appropriate synchronous line protocols, DIGITAL's own DDCMP, for interfacing to other DIGITAL products, and BISYNC for interfacing to non-DIGITAL products are supported.

Users may configure networks with simple or complex topologies utilizing several families of communications front-end processors, remote batch stations, remote terminal concentrators, and combination remote batch/terminal concentration products. These products feature functionality throughout a wide spectrum of communication needs.

Additional details concerning TOPS–10-based communication products are discussed in Section 10 of this document.

RELIABILITY

Built-in reliability features for both hardware and software provide data integrity, increased uptime, and fast system recovery from power, hardware, or software failures. Some of the reliability features are discussed in the following paragraphs.

System Availability

The TOPS–10 operating system allows the hardware system to continue running even though some of the hardware components have failed. The system automatically determines the presence of peripheral devices on the system when the system is started. If the usual system bootstrap device is unavailable, the system can be bootstrapped from another disk drive or from magnetic tape. If memory units are defective, memory is configured so that defective modules are not referenced. Software spooling allows output to be generated even if the normal output devices are not available.

The system operator can perform software maintenance activities without bringing the system down for stand-alone use. The operator can perform disk backup and restore procedures for all files on the system or for just a single file concurrent with normal activities.

The TOPS–10 operating system supports on-line peripheral diagnostics. TOPS–10 performs on-line logging of CPU errors, peripheral device errors, and software failures. The operator or field service engineer can examine and analyze the error log file while the system is in operation.

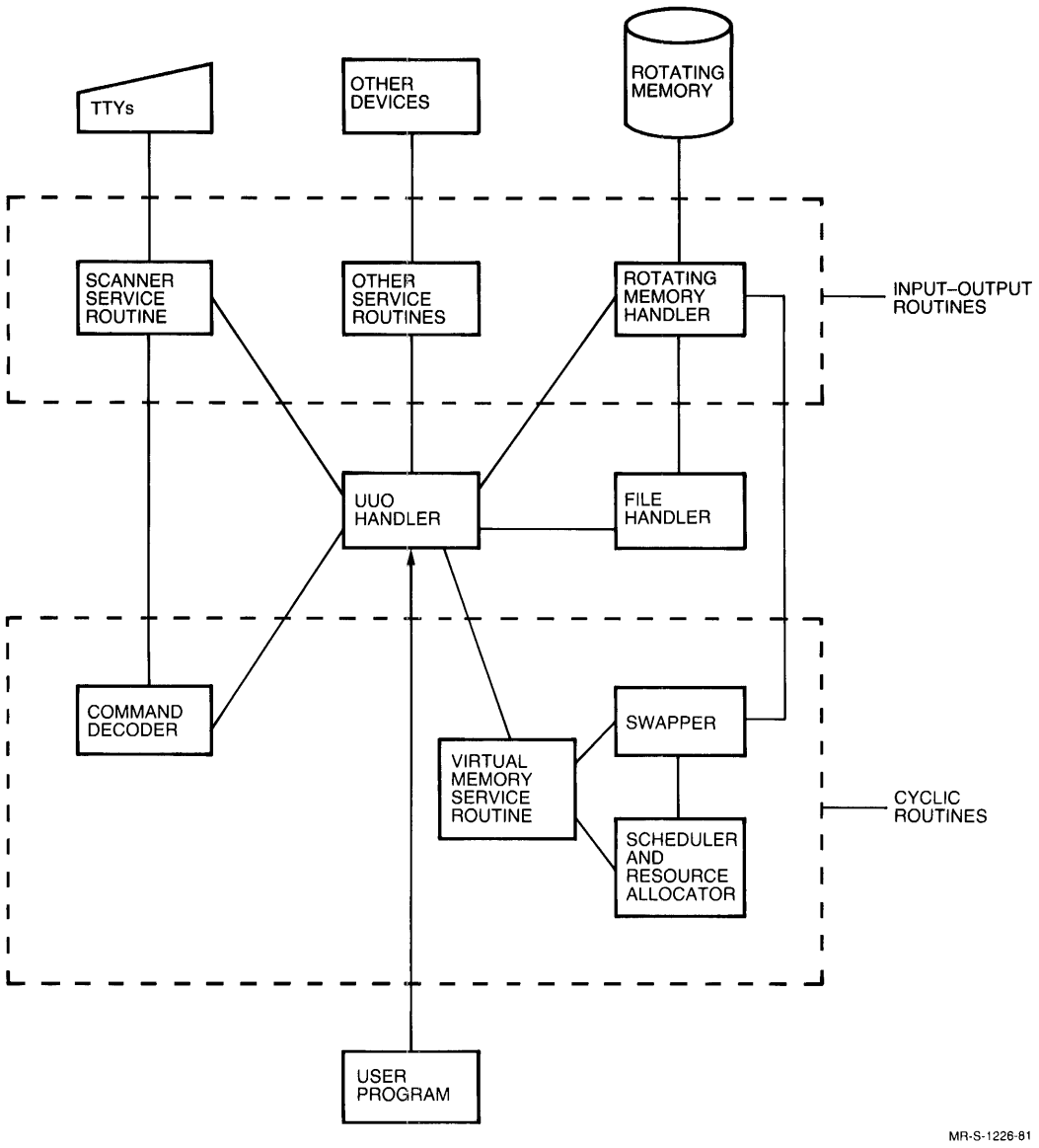


Figure 2–1
Interrelationship of TOPS–10 Modules

Availability Reporting

The TOPS–10 operating system maintains a disk file containing the times and reasons for system reloads. A program can be run periodically to read this file and generate reports on system availability. These reports provide information pertaining to the overall system availabilty and probability of completing jobs of certain duration.

System Integrity

If system power fails, a power failure detection circuit senses the condition and causes an interrupt. The interrupt triggers the operating system to save all valuable registers so that the system can be restarted in a minimum amount of time.

On the KL10, through the PDP–11 console computer, an automatic restart capability has been added to resume normal operations in the event of a power failure. All three phases of AC power are monitored. Low voltage on any phase initiates a sequence of power-down operations. A program-selectable automatic restart capability is provided to allow resumption of operations when power returns. Alternatively, a manual restart may be used.

Temperature sensors strategically placed within the system, detect high-temperature conditions and cause power shutdown. This, in turn, initiates the power failure interrupt.

System Recovery

Automatic system restart facilities attempt to bring up the system without operator intervention after a system failure caused by a power interruption, a machine check hardware malfunction, or a fatal software error. TOPS–10 automatically performs machine checks and internal software consistency checks during system operation.

TOPS–10 remote diagnosis allows DIGITAL Field Service Engineers to run diagnostics, examine memory locations, and diagnose problems from a remote terminal. The field service engineer who goes to the site is prepared in advance to correct any problems that might have occurred.

Error Reporting

TOPS–10 provides an extensive error detection and recovery package to ensure maximum system availability to the user. It also provides complete error reporting facilities for DIGITAL Field Service and Software Support personnel, and the customer's operation staff.

When an error is detected, TOPS–10 gathers all pertinent hardware and software information, including whether the error is recoverable, invokes the recovery procedure, and adds this information to a disk file for storage. Later, a program (SYSERR) may be run to read this file and generate reports concerning the entire system or individual items. Additionally, significant operational events, such as system reloads and changes in the system configuration, are recorded to assist the operation staff in monitoring system performance.

On a periodic basis field service engineers gather summary information from the error log file and can detect indications of potential problems. Detailed reports about specific errors frequently allow diagnosis of a problem without having to run exhaustive diagnostics in stand-alone mode.

The error file can be backed-up on magnetic tape to provide a complete history of system operation and quickly pinpoint slowly degrading portions of the system long before serious problems occur.

3
The Users

The DECsystem-10 consists of hardware and software that allows the user to run a variety of programs efficiently and conveniently. The system as a whole is designed to execute many different kinds of jobs concurrently.

The DECsystem-10 provides a complete program development environment. In addition to the native assembly language, it offers optional high-level programming languages commonly used in developing scientific and commercial applications. It also offers the implementation language BLISS–36 for system programming applications. The DECsystem-10 provides the tools necessary to write, assemble, or compile and link programs, as well as build libraries of source and object modules.

The system's users are those people who interact with applications or system jobs at an on-line terminal, or who benifit from production batch jobs. These users can control the operation of the system through the TOPS–10 Operating System command language. This command language is used by system programmers to develop application software, by operators to monitor the system, and by system managers to control resources of the system.

The TOPS–10 operating system takes maximum advantage of system throughput capabilities, allowing many independent users to share system facilities simultaneously. Its rapid-response nature makes it particularly well suited for a wide range of tasks.

This section looks at the system from four different usage perspectives:

- Application Programmers
- System Programmers
- System Managers
- System Operators

THE APPLICATION PROGRAMMER

The application programmer can write, compile, edit, and test programs both interactively and in batch mode. As an interactive user, the application programmer can control the running of a program; create, edit, and delete data; compile, execute, and debug programs; request assignment of peripheral devices such as magnetic tapes; and make use of all system features from a terminal. As a batch user, the application programmer can submit a job into the system using a card deck that contains control cards defining the command options and error recovery procedures for the job; or, using a terminal, the user can create and submit a control file which is then interpreted by the batch system and processed in exactly the same manner as the job submitted on cards.

Programmers can use the system for development while other user jobs are in progress. Programmers can interact with the system on-line, execute command procedures, or submit command procedures as batch jobs.

All DECsystem-10 languages and compilers are shared and reentrant, allowing many programmers to work simultaneously using the same copy of various compilers.

The Programming Languages

The following is a list of languages offered by DIGITAL.

- COBOL is the recognized "big machine" business language. DECsystem-10 COBOL provides extensive data processing capabilities for commercial applications. TOPS–10 COBOL is written to the ANSI 68 and 74 specification, so it is compatible with the COBOL used on most general purpose systems. Both COBOL–68 and COBOL–74 provide an interactive debugger, a source copy library utility, ISAM utility, and a report writer facility as standard tools.
- FORTRAN is the standard scientific language. DECsystem-10 FORTRAN–10 consists of a globally optimizing compiler and runtime system with interactive debugger providing fast program development and execution.
- ALGOL is a sophisticated scientific programming language. ALGOL–10 is a one-pass, single-phase compiler that generates optimized object code.
- APL (A Programming Language) is a concise programming language suitable for manipulating numeric and character-oriented array-structured data. It includes procedural operators for array calculations and its own editing and debugging facilities.
- CPL (Conversational Programming Language) is an interpreter supporting a subset of the ANSI PL/1 language.
- BASIC is a problem-solving language that is easy to learn because of its conversational nature. It is particulary well suited to a timesharing environment because of the ease of interaction between the user and the computer. BASIC can be used to solve problems with varying degrees of complexity, and thus has a wide application in the educational, business, and scientific markets.
- BLISS–36 is DIGITAL's implementation language for system development. BLISS–36 is an optimizing, high-level systems implementation language for the DECsystem-10. It is specifically designed for building compiliers, real-time processors, utilities, and operating system software. BLISS encourages the writing of highly structured programs that are easy to maintain.

Application Tools

The following are just a few of the tools available to the application programmer.

Database Management

The TOPS–10 Data Base Management System (DBMS) satisfies the need for high-performance, large volume data management. Conceptually, DBMS–10 provides four capabilities:

- CODASYL implementation
- Centralized data definition
- Multithreaded data manipulation
- Utilities for effective maintenance and administration

Data Definition

The central definition of the data base allows access by many applications written in different languages. Using the Data Definition Language (DDL), the data base administrator defines the logical and physical characteristics of the data base files and data base records. BY controlling the placement of records on the physical storage medium, the administrator can assure both data security and high-speed data retrieval.

Records can be put together in logical groups called "sets". Sets are used to model the real-world relationships among objects; instead of storing extra data about relationships, DBMS–10 links related records in logical chains.

The data base definitions reside in a central location in the Schema. User programs can access these definitions through Sub-Schemas. A Sub-Schema is a subset of the schema that is defined by the administrator to satisfy the needs of related groups of programs. By restricting the scope of the Sub-Schema, the administrator can prevent certain classes of users from accessing portions of the data base, from the level of files down to the level of data items. This feature provides one of the inherent security checks of DBMS–10.

Data Manipulation

Data base retrieval and updating is performed by the Data Manipulation Language (DML). The programmer includes DML statements in a COBOL or FORTRAN program. These two languages are called the "host languages" for the DML.

The first DML statement a program contains is the INVOKE statement. This statement provides the program with the definitions from the Sub-Schema, in a form suitable for the host compiler. No further data declarations are needed in order to access the data base items; the centrally defined data base structures are mapped directly into the host program.

DBMS–10 provides multithreaded simultaneous update. Many users can access the data base at the same time. Each program is protected by a system of locks that are controlled in the Schema. The data base administrator can grant exclusive control to one program, or distribute control among many programs. When a large number of programs are using the data base, each one locks the resources it is using until it completes its transaction.

Two typical DML verbs are FIND and STORE. In DBMS–10, there are six different ways to find a record. A record can be found by traversing a set, by hashing a data item, by sorted order, by direct address, by relative address, or by current program context.

When a record is read in from disk, DBMS–10 can also read in other data base pages containing related records. This is one of the many ways in which DBMS–10 takes advantage of the TOPS–10 operating system to maximize the efficiency of your program.

The STORE statement, in addition to placing the record in the data base, automatically performs a number of operations. In accordance with the specifications of the Schema, a record can be stored next to its logical group, or it can be distributed randomly across the data base. In addition, the STORE operation can automatically choose an appropriate set and link the record into it immediately.

An important but largely invisible component of DBMS–10 is the journal. A journal is a record of everything that has happened to the data base. When a program is using the journal, the program can recover from any kind of error that affects the data base. The data base is restored to the state it was in before the current transaction began. This can be done either automatically or under program control. The journal is an important feature that assures data base consistency, integrity, and reliability.

Data Base Utilities

DBMS–10 includes three utility programs that aid in the design, administration, and maintenance of your data base.

DBMEND is a utility for repairing your data base if anything goes wrong. Using the journal, DBMEND can restore the data base after a system software or hardware failure, or a user error.

DBINFO generates reports about the structure and contents of the data base. A variety of reports are available, including maps and dictionaries from the Schema. Users can also obtain data dumps of single pages, single sets, single records, or any larger portion of the data base.

The STATS utility is a subroutine that can be called by a data base program. This utility generates complete statistics about the performance of the data base. These statistics, relating to buffer management, disk I/O, lock performance, and DML runtime, allow the data base administrator to evaluate the data base design and to fine-tune it for maximum performance.

Message Control System (MCS–10)

The Message Control System (MCS–10) is a facility of TOPS–10 that allows an installation to control communications between a network of terminals and application programs.

This communication is different from timesharing because the terminal user never logs onto the system

and never runs his own job. The user simply sends data and receives an answer. When MCS–10 receives data from a terminal, it passes the data to a COBOL program for processing. The program, after it receives the input data, sends a reply to the terminal through MCS–10.

The COBOL programs that process the transactions are called Message-Processing Programs (MPPs). Each MPP, when processing a transaction, runs as a subjob of MCS–10 through a resource called a job slot. The group of terminals that transmit and receive messages is called a network.

When MCS–10 receives transactions from the terminal, it stores them in input queues where they wait until MPP receives them. When the MPPs send transactions back to the terminal, MCS–10 places the transactions into output queues, which are then accessed by the terminal.

THE SYSTEM PROGRAMMER

The system programmer uses the system to design and develop application systems for multiprogramming environments requiring fast response and a high degree of job interaction and data sharing.

System programmers are provided with a range of tools for the development of high-level languages, system utilities, operating systems, and highly sophisticated applications where manipulation of hardware is required.

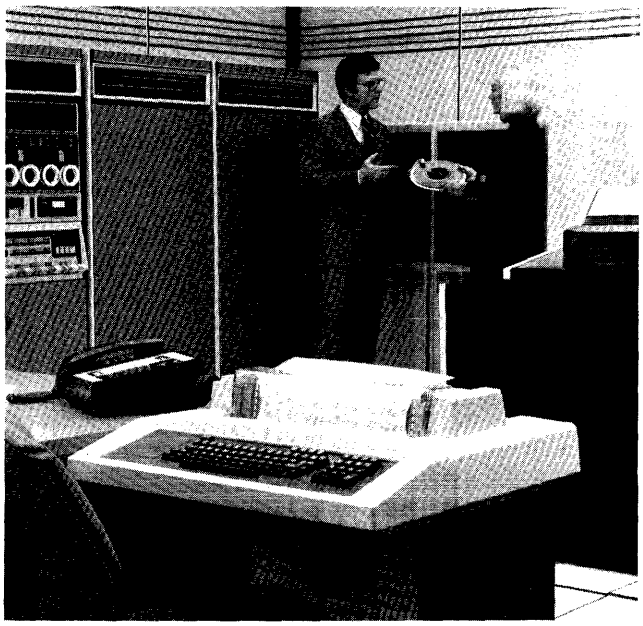
System programmers use MACRO, the DECsystem-10 symbolic assembly language. MACRO makes machine language programming easier and faster by translating symbolic operation codes in the source program into binary machine language instructions. MACRO relates symbols that are specified by the user to stored numeric values. It assigns relative memory addresses to symbolic addresses of program instructions and data.

System programmers may also use BLISS–36 for software development. BLISS–36 is an optimizing high-level system implementation language. It is specifically designed for building compilers, system utilities, and operating system software.

System programmers applications normally communicate with the system through monitor calls. Monitor calls are used to request system functions, such as input/output operations, during execution of the program. Support programs, such as the COBOL compiler, issue requests to the system through monitor calls. All monitor calls are reentrant. This maximizes efficiency by allowing several jobs to be in process at the same time and maximizes system reliability by isolating monitor call processing from other operating system processing.

THE SYSTEM MANAGER

The system manager has the responsibility for planning data access and protection, granting privileges, authorizing system use, controlling resource utilization, and analyzing the system's accounting and performance information.



User Authorization

The system manager controls use of the system primarily by creating user directories. The user directories are used to identify the user, supply defaults, specify privileges, and limit resource usage.

The system manager assigns a Project-Programmer Number (PPN) and password to each user who wishes to access the system. When the user logs on to the system, the system verifies the user's PPN and password. If either the PPN and/or password are incorrect, the system refuses the user access to the system. This feature prevents unauthorized use of the system.

Privileges

The system manager can assign specific users special privileges. These privileges include interprocess communications and control, performance control, file and device access, and system operational control privileges.

If a user tries to execute a function that requires a specific privilege, the system checks to see if he is allowed to use that privilege. If the user does not have the specific privilege, the system does not execute the function and notifies the user that he does not have privileges.

Allocating Disk Storage Quotas

The system manager assigns each directory a specific number of blocks for both working storage and permanent storage. Working storage refers to the disk space that a user can have during the time he is logged on the system. Permanent storage refers to the total disk space that the user can have to store files after he has logged off the system.

Storage allocations are strictly enforced. Users cannot exceed their working storage allocation unless they have certain privileges. If a user tries to create a file and he is over his working storage allocation, the system does not let him create the file and notifies him that he is over his working storage allocation.

Controlling Resources

The system manager can make policy decisions that govern the access to a specific system resource. For example, TOPS–10 allows a user to assign a device, log on to the system at any time of day, mount a magnetic tape, and mount a disk structure. The access control feature allows the system manager to restrict or disallow the use of some of these facilities. The system manager can allow only certain users at specified times of day and, perhaps, at specified terminals, to use certain facilities. By using the access control mechanism, the system manager can reduce or prevent malicious access to the system resources, and has an additional means for collecting accounting or other types of information.

Scheduler Controls

The system manager is provided with a tool that allows him to control or fine-tune the allocation of CPU time on the basis of classes. This tuning mechanism is called class scheduling.

Class scheduling allows the system manager to allocate percentages of CPU time to individual classes of users. Each job in a class receives a portion of the class percentage. By using the class scheduler, the system manager can provide a consistent service to predefined groups of users. The system manager can also set up a class to include all batch jobs. This allows the system manager to control batch jobs separately from interactive jobs.

The system manager can change the percentage of a specific class while the system is in operation. This change in percentage remains valid until the system is reloaded or until another change is made.

Resource Accounting Statistics

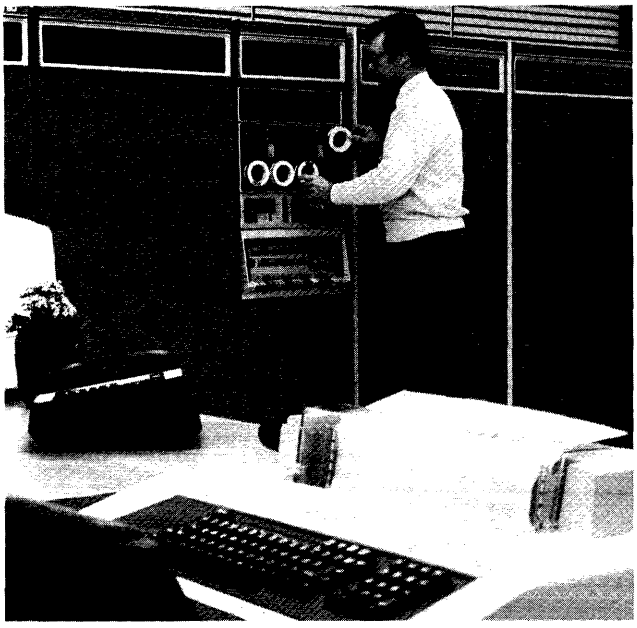
The system manager is provided with an accounting facility that allows him to assign and charge computer usage to user accounts. This feature provides the system manager with the means to add security to the system, determine charges for computer usage, and bill users by account name.

All accounting data is stored in a file and can be used later for reports and billing. Because the system collects all detail records, system managers can define their own algorithms for resource usage billing.

Performance Analysis Statistics

The system manager is provided with a program that collects data about system usage and performance. This program aids the system manager to tune the system for maximum performance. Some of the statistics gathered by the program are listed below.

- Monitor Statistics — indicate CPU, disk, memory utilization, and system performance
- Job Statistics — indicate how much CPU time each job used, information on each job's use of system resources, and a summary of statistics collected on the individual jobs
- System Utilization Statistics — indicate the demands made on the system and the distribution of system resources
- Disk I/O Statistics — indicate the number of seeks, reads, and writes performed by each disk drive



THE SYSTEM OPERATOR

The system operator has complete control of the system. The system operator is responsible for preparing the system for timesharing and batch work, for responding quickly to user requests such as magnetic tape mounts, for taking care of the line printer and the distribution of printed output, and for recovering the system when errors occur. The system operator may be called upon by the system manager to perform a variety of other functions, both hardware and software related, to ensure the efficient running of the system.

Operator Interface

The TOPS–10 operator interface is called OPSER (OPerator SERvice) OPSER facilitates the operations of the jobs/system programs that the operator needs for efficient operation of the system. Under OPSER, the operator can process auto files, create subjobs, list the available resources of the system, restrict certain system resources to operator's use, send messages to all active terminals, restrict the number of jobs running on the system or the maximum memory used by jobs, run system status displays, and receive messages from users or from the system.

Controlling Batch Streams

The system operator can control the number of batch job streams can run concurrently. Batch jobs can be submitted by an interactive user, another batch job, or any program. When the number of batch jobs exceed the number of batch streams, the remainder of the batch jobs are held in a batch input queue. The system operator can dynamically start additional batch streams when it is required. The operator can also stop, continue, and shut down batch streams. The operator can send messages to batch streams, set parameters for batch streams, and display status and parameters of batch streams.

System Recovery

The system operator can select manual or automatic system recovery following a power interruption or a hardware or software failure.

Using automatic system recovery after a power interruption, the system determines whether the contents of memory are still valid, and, if so , restarts all I/O that was in progress at the time of interruption and continues operation from the point of interruption. If the contents of memory are not valid, the system automatically restarts itself from disk and executes start-up command procedures.

If the normal system's bootstrap device is unavailable, the operator can boot the system from several different types of devices.

System Backup

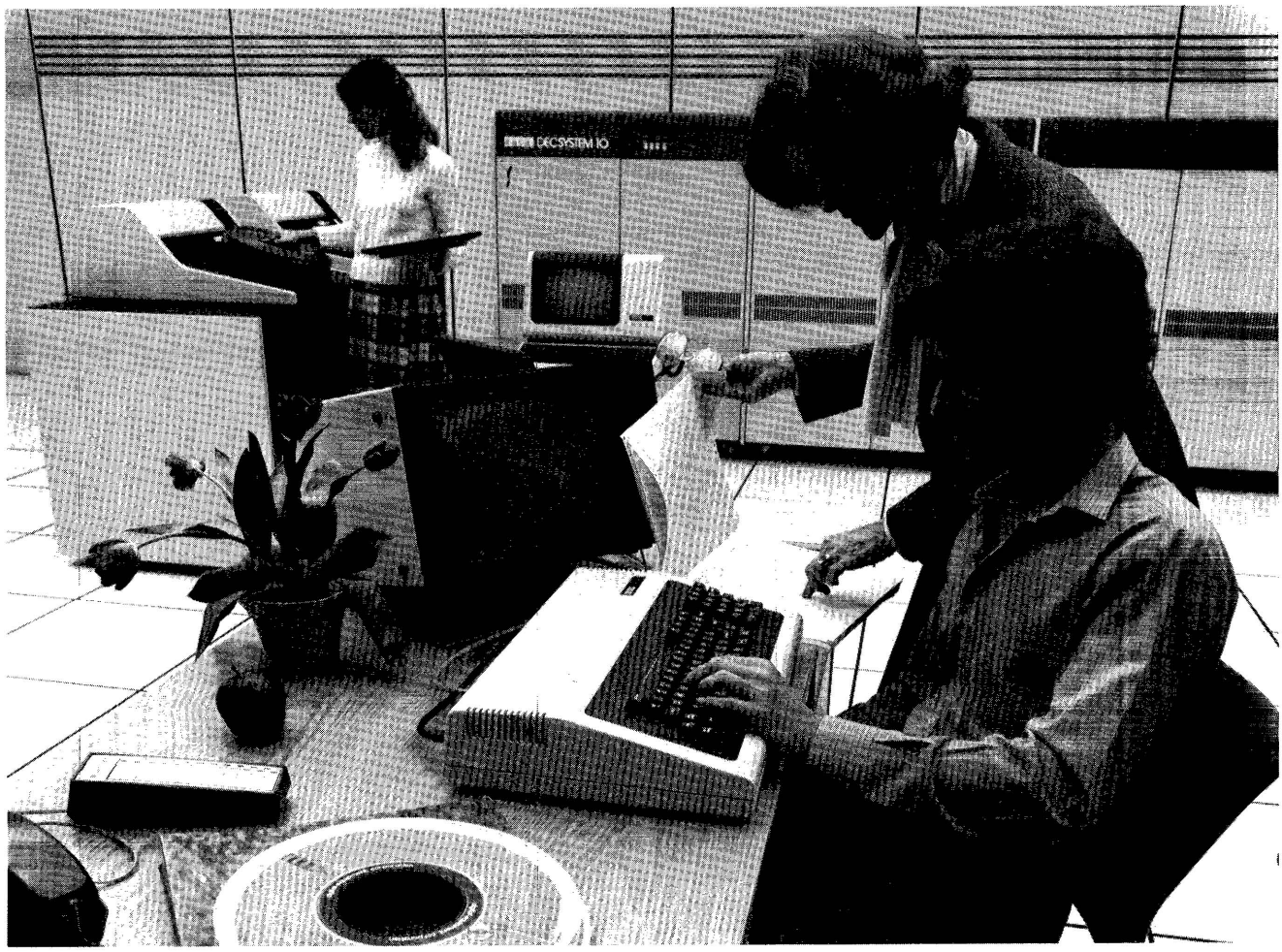
The system operator is provided with a facility that can be used to save all files or selected files on tape. The operator can then use this tape to restore all or certain files back on to disk if needed.

USER UTILITIES

The TOPS–10 operating system provides users with several utility programs that aid the user in performing various tasks. The following is a list of the most frequently used utilities.

- CREF program allows the user to make a cross-reference listing of symbols used in MACRO and FORTRAN programs.
- FILCOM program allows the user to compare two files on a line-by-line basis and list the differences.
- BACKUP program allows the user to copy disk files to magnetic tape for safekeeping and/or to transfer files between systems. BACKUP also allows a full range of functionality including incremental saving of a single file or all files on the system.
- LINK program allows the user to merge independently translated program modules and system modules into a single module that can be executed by the operating system.
- DDT program allows the user to examine, search, change, insert breakpoint instructions, and stop/trace a program at symbolic level.
- PIP program allows the user to transfer files between standard I/O devices and can be used to perform simple editing and magnetic tape control operations during those transfer operations.
- RUNOFF program allows the user to easily prepare documents in conjunction with a text editor or batch system.
- TECO program allows the user to create and modify character-oriented programs and data files on-line.
- REACT program allows the designing and maintaining administrative control files.

4
The Operating System



The TOPS–10 operating system provides a highly reliable multipurpose, multiprogramming, and multiprocessing operating system. As a multipurpose operating system, it serves interactive timesharing and performs multiprogram batch processing simultaneously. TOPS–10 reacts to inquiries, requests, and demands from many different users at local and remote stations; it is able to store, retrieve, and protect large blocks of data; and it also makes optimum use of the available hardware facilities, while minimizing turnaround time.

TOPS–10 is designed for the concurrent operation of interactive timesharing multistream batch, real-time, and remote communications in either single or multiprocessor system configurations. In providing these multifunction capabilities, TOPS–10 services interactive users, operates local and remote batch stations, handles multiple transaction- oriented terminals, and performs data acquistions and control functions for on-line laboratories and other real-time projects. By dynamically adjusting system operation, the TOPS–10 operating system provides many features for each class of user and is therefore able to meet a wide variety of computational requirements.

Jobs running under control of TOPS–10 are similarly configured, so that only active jobs occupy main memory. The remaining jobs stay on mass storage devices so that the system resources are used with maximum efficiency. TOPS–10 swaps jobs in and out of main memory dynamically. The job is always ready when the processor calls for it. Dynamic relocation is handled jointly by hardware and TOPS–10, and does not require intervention from the user's program.

The TOPS–10 operating system has been designed to handle a wide variety of applications in a simple, flexible, and efficient manner. It includes:

- Interactive and batch processing
- Real-time computing
- A flexible file system
- A common command language for both interactive and batch processing
- The scheduler
- The swapper
- The UUO handler
- Input/Output routines

INTERACTIVE TIMESHARING

TOPS–10 takes maximum advantage of the capabilities of the hardware system by allowing many independent users to share the facilities of the system concurrently. Because of the interactive, conversational, rapid response nature of TOPS–10, a wide range of tasks — from solving simple mathematical problems to implementing complete and complex information gathering and processing networks — can be performed by many users at the same time, each with the impression that the entire computer is servicing his needs. The number of users on the system at any one time depends on the system configuration and the mixture of jobs on the system. TOPS–10 timesharing is designed to support in excess of 175 active users. Interactive terminals can include CRTs, hard-copy terminals, and other devices that operate at speeds from 110 to 9600 baud (asynchronous lines). Terminal users can be located at remote locations connected to the computer center by communication lines using modems.

Timesharing users can spool output to a line printer. This feature allows the system to optimize use of the device by sharing it among all users. Timesharing also allows control over an output file by allowing the user to specify the characteristics of the device and lets the system determine which physical device can meet the users need.

Timesharing under TOPS–10 is designed so that the command language, input/output processing, file structures, and job scheduling are independent of the programming language used. In addition, standard software interfaces make it easy for the user to develop special languages or systems. This general purpose approach is demonstrated by the variety of special purpose programming languages implemented under TOPS–10.

BATCH PROCESSING

The TOPS–10 batch subsystem enables users to execute batch jobs concurrently with timesharing jobs. The batch user communicates with the system in the same command language as a timesharing user. Because TOPS–10 runs batch and interactive processing simultaneously, a user can debug a program under timesharing and then run the program under batch, without any additional coding.

The batch system consists of several components, as shown in Figure 4–1. They are:

- The input spooler
- The queue manager
- The scheduler
- The output spooler

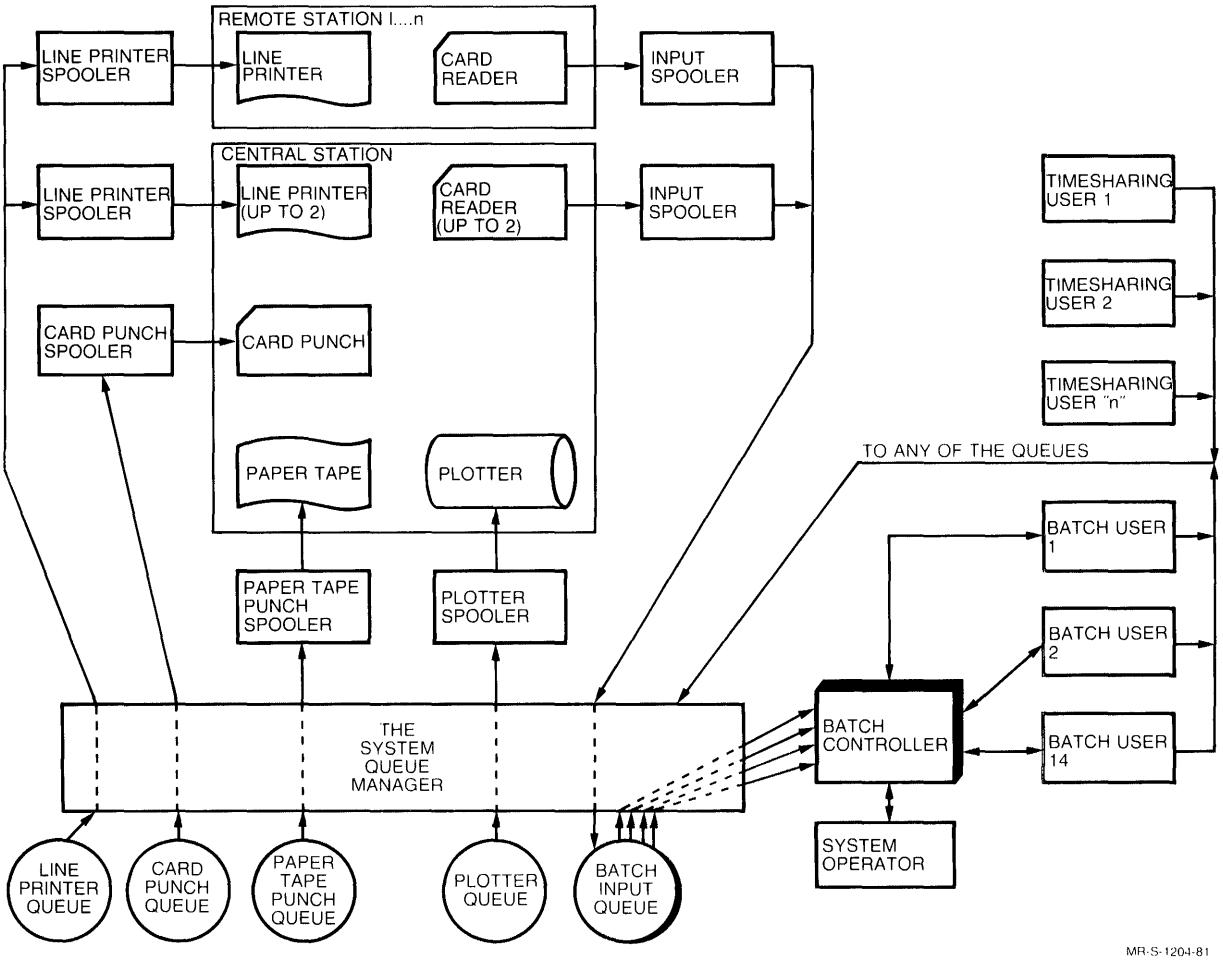


Figure 4–1
GALAXY Batch Software

The Input Spooler

A user has two methods of submitting a job into the batch system. The first method is by creating a control file on a terminal. This control file contains instructions to the operating system as well as instructions to the programs to be run by the batch job. The control file can also contain instructions to the batch controller to make decisions and possibly jump over instructions.

The second method of submitting a batch job is from the card reader. In this case, the card deck constitutes the control file. When cards are used, the input spooler reads the cards and stores the information in a control file on disk. The action of reading cards continuously and storing the information onto disk as files until the system processes the information is called input spooling. Cards can be read from a card reader directly attached to the main system, or through a remote job entry station (RJE) attached to the main system by a communication line.

Each control file constitutes a job and is placed in a priority queue. A priority queue is an arrangement of jobs in an ordered list. The jobs are inserted into the list according to their priority. The higher the priority of the job, the higher its position in the run queue.

The Batch Scheduler

When the input spooler queues a job, it notifies the batch scheduler. The batch scheduler chooses one job from those that have been spooled to disk. This choice is made according to the job's priority. Under TOPS–10, the priority of a batch job is based on the priority set by the user at the time the job was submitted, the type of batch streams which can process the job, and the size of the job. Jobs submitted on cards have a fixed priority, but those submitted from a terminal have assignable priorities. If no special priority is indicated, all batch jobs have equal priority, which means that the queues are in a first-in-first-out order. Only two things can disrupt this order: assignment of a higher priority, and accumulation of wait time in the queue due to other jobs moving ahead.

When choosing a job, the batch scheduler must be sure sufficient memory is available, and must determine what other resources are needed to start the job. Several batch jobs may be scheduled simultaneously depending upon available resources.

The Queue Manager

In TOPS–10 the queues used by the batch scheduler are built and maintained by the queue manager. When a job is chosen, the queue manager makes an entry in the batch controller's input queue based on its priority. The batch controller then processes the job and notifies the queue manager when it has completed the job. After the job is completed, the queue manager schedules the job for output by placing an entry in the output queue.

The Output Spooler

When the output spooler receives a request for output, it performs the output procedure and notifies the scheduler when it is finished. This output can include a log file created by the input spooler when the job was submitted. The log file indicates the time of each significant event in the processing of the job. When the output has been processed, the queue manager deletes the entry from the output queue. Interactive users use this output queue when they give various commands, such as PRINT, ULIST and others.

Flexibilty

The batch system offers tremendous flexibilty. Card jobs normally come from the card reader, but can originate on disk. The latter facility allows the programmer to develop and debug batch jobs in interactive mode. In both instances, the batch system preprocesses the job before it is queued. A batch job can also be entered from an interactive terminal, in which case the user bypasses the input spooler and creates a control file directly on disk. The control file contains operating system and batch commands along with the user program commands necessary to run the job. The user then enters the job into the batch system by way of an operating system command string. The user can include switches to define the operation and set the priorities and limits on memory and processor time.

Job Dependency

Although jobs are entered sequentially into the batch system, they are not necessarily run in the order that they are read because of priorities either set by the user in a control card or operating system command, or computed by the queue manager when determining the scheduling of jobs. Occasionally, the user may wish to submit jobs that must be run in a particular order. In other words, the execution of one job is dependent on the sucessful execution of a previous job. To ensure that jobs are executed in the proper sequence, the user must specify an initial dependency count when submitting the dependent job. The dependency count is then part of the input queue entry. A control command in preliminary job decrements the count. When the count becomes zero, the dependent job is executed.

Error Recovery

The user can control system response to error conditions by including commands to the batch controller which aid in error recovery. With error recovery commands, the user specifies the action to be taken when his program contains a fatal error. For example, if a program contains a fatal error, the user may specify

that the batch controller cause a jump to the next program or jump to a user-written error-handling routine. If an error occurs and the user did not include error recovery conditions in his job, the batch controller initiates a standard dump of the user's program space and terminates the job. This memory dump provides the user with the means to debug his program. If the system fails for some reason and is restarted, the job continues at a point prespecified by the user.

Although the batch system allows a large number of parameters to be specified, it is capable of operating with very few user-specified values. If a parameter is missing, the batch system supplies a reasonable and predictable default value. These defaults can be modified by the individual installation.

Operator Intervention

Normal operating functions performed by a program in the batch system require little or no operator intervention. However, the operator can exercise a great deal of control if necessary. The operator can specify the number of system resources to be dedicated to batch processing by limiting the number of batch jobs that can be run on the system simultaneously. He can prevent a job from being processed, then release it from this hold; stop a job in progress, and then requeue it; change a job's priorities; or abort a job in progress. By examining the system queues, he can determine the status of all batch jobs.

In addition, the programs in the batch system can exchange information with the operator. The user's batch job itself may also communicate with the operator. All operator intervention occurring during specific operations causes messages to be written in the user's log file, as well as in the operator's log file, for later analysis.

REAL–TIME COMPUTING

For a system to be satisfactory for real-time operations, two important requirements must be met. The most important requirement is fast response. Because real-time devices may not be able to store information until the computing system is ready to accept it, the system would be useless for real-time if the response requirements of a real-time process could not be satisfied. The TOPS–10 operating system allocates system resources dynamically in order to satisfy the response and computational requirements of real-time jobs.

The second requirement is protection. Each user of the system must be protected from other users, just as the system itself is protected from all user program errors. In addition, because real-time systems have special real-time devices associated with jobs, the computing system must be protected from software and hardware failure that could cause a system crash.

Inherent in the TOPS–10 operating system is the capability of real-time processing, and it is by way of monitor calls (UUOs) to the operating system that the user obtains real-time service. The services obtained by monitor calls within the user's program include:

- Locking a job in memory
- Connecting a real-time device to the priority interrupt system
- Initiating the execution of FORTRAN or machine language code on receipt of an interrupt
- Disconnecting a real-time device from the priority interrupt system

Locking Jobs

Memory space is occupied by the resident TOPS–10 operating system and by a mix of real-time and non-real-time jobs. The only fixed partition is between the resident operating system and the remainder of memory.

To assure that information is not lost when the interrupt is received from a real-time device, and the user program necessary to service that data is not in main memory, the job can request that it be locked in memory. This means that the job is not to be swapped to secondary storage and guarantees that the job is available when needed, without waiting for swapping delay. Because memory is not divided into fixed partitions, but rather dynamically assigned, it can be utilized to a better degree by dynamically allocating more space to real-time jobs when real-time demands are high. As real-time demands lessen, more memory can be made available to regular timesharing and batch usage by unlocking the real-time jobs from memory-resident status.

Real-Time Devices

With the KL10, the real-time user can connect real-time devices to the priority interrupt system, respond to these devices at interrupt level, remove the devices from the interrupt system, and/or change the priority level on which these devices are assigned. There is no requirement that the devices be connected at system installation time. The user specifies both the names of the devices generating the interrupts and the priority levels on which the devices should function. The operating system links the devices to the appropriate interrupt level modules within itself.

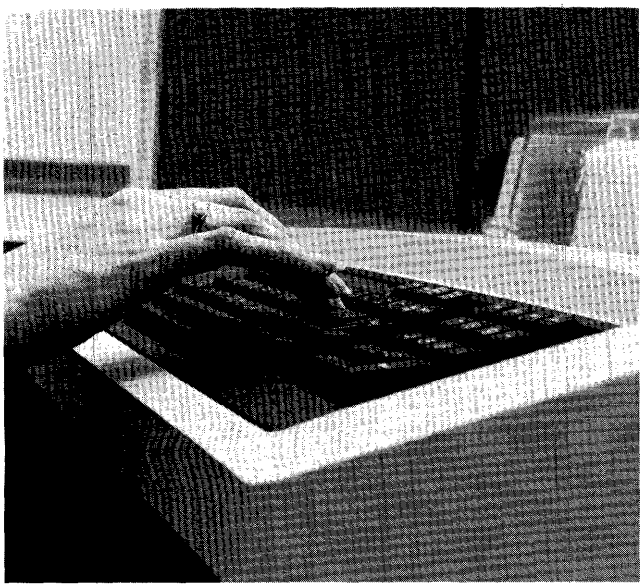
A user can control the real-time device in one of two ways: single mode or block mode. In single mode, the user's interrupt program is run every time the real-time device interrupts. In block mode, the user's interrupt program is run after an entire block (defined by the user) has been read from or written to the real-time device. When the interrupt occurs from the device in

either single mode or at the end of a block of data in block mode, the operating system saves the current state of memory and services to the user's interrupt routine. The user services his device and then returns control to the operating system to restore the previous state of the system and to dismiss the interrupt.

Any number of real-time devices may be placed on any available priority interrupt channel. When a prespecified event occurs, the dormant program is activated to process the data. The main memory space for the real-time job's buffer does not need to be reserved at system installation time. The hardware, working in conjunction with the operating system's memory management facilities, provides optimum main memory usage.

High-Priority Run Queues

The real-time user can receive faster response by placing jobs in high-priority run queues. These queues are examined before all other run queues in the system, and any runnable job in a high-priority queue is executed before jobs in other queues. In addition, jobs in high-priority queues are not swapped to secondary storage until all other queues have been scanned for a swappable candidate. When jobs in a high-priority queue are to be swapped, the lowest priority job is swapped first and the highest priority job is swapped last. The highest priority job swapped to secondary storage is the first job to be brought into memory for immediate execution. Therefore, in addition to being scanned before all other queues for job execution, the high-priority queues are given greater consideration in the swapping decision.



COMMAND LANGUAGE

By allowing resources to be shared among users, the TOPS–10 multitask, multiuser environment is able to efficiently use valuable system resources that are wasted or inefficiently used in a single-user system. Users are not restricted to a small set of system resources, but instead are provided with the full variety of facilities. By means of an interactive terminal, a user has on-line access to most of the system features. This on-line access is available through the operating system command language. Through the use of this language, the interactive user communicates with the operating system. Thus, the user is able to completely control the running of a task, or job, to achieve the desired results: create, edit, and delete files; start, suspend, and terminate a job; compile, execute, debug, and run programs. In addition, because the batch software system accepts the same command language as the interactive software, any user can enter a program into the batch run queue without complex control information or JCL. This allows any interactive terminal to act as a remote job entry terminal.

When a user types commands and/or requests on his terminal, the characters are stored in an input buffer in the operating system. The command interpreter examines these characters, checks them for correct syntax, and invokes the system program or user program as specified by the command string.

On each clock interrupt, control is given to the command interpreter to interpret and process one command in the input buffer. The command appearing in the input buffer is matched with the table of valid commands accepted by the operating system. A match occurs if the command typed matches a command stored in the system, or if the characters typed match the beginning characters of only one command (that is, constitute a unique abbreviation). When this match is encountered, the legality information (or flags) associated with the command is checked to see if the command can be performed immediately. For instance, a command must be delayed if the job is swapped out to secondary storage and the command requires that the job be resident in main memory; the command is executed on a later clock interrupt when the job is back in memory. If all conditions as specified by the legality flags are met, control is passed to the appropriate program.

The user can also request assignment of any peripheral device (magnetic tape, DECtape, and private disk pack) with the command language. When the request for assignment is received, the operating system verifies that the device is available for the user; and, if so, the user is granted the device for his private use until he relinquishes control. In this way, the user can also

have complete control of devices such as card readers and punches,and line printers. This private use feature is under the complete control of the system operator. In some installations, sharable peripheral devices are maintained in an resource "pool" from which a user requests exclusive use of devices. The operator then assigns the requested device to the user.

THE FILE SYSTEM

Under typical TOPS–10 operation, mass storage devices, such as disk, cannot be requested for a user's exclusive use, but must be shared among all users. However, if system capacity allows and a user has proper authorization, TOPS–10 allows a user to mount and assign a disk structure for private use. Because many users share the system's mass storage devices, the operating system must ensure independence among the users; one user's actions must not affect another user's activities unless the users want to work together. To guarantee such independence and security, the operating system provides a file system for disks. Each user's data is organized into groups of 128-word (36-bit) blocks called files. The user gives a name to each of his files and the list of these names is kept by the operating system for each user. The operating system is then responsible for protecting each user's file storage from intrusion by unauthorized users.

In addition to allowing independent file storage for users, the operating system permits users to share files. For example, programmers working on the same project can share the same data in order to complete a project without duplication of effort. TOPS–10 allows the user to specify access rights (protection codes), for his own files. These codes designate whether other users may read the file, and, after access, if the files can be modified in any way. A facility called File DAEMON allows the user to specifically permit or deny access to any file or set of files for specific users. The user may also create a log of accesses to his files for later review, proprietary program billing, project maintenance, and others using this facility.

The TOPS–10 user is not required to preallocate file storage; rather, the operating system allocates and deallocates the file storage space dynamically upon demand. Not only is this convenient for the user because he never has to calculate necessary storage for a given program, but this feature also conserves storage space by preventing large portions of storage from being waited. However, a large job that needs to preallocate file storage space for efficiency may do so.

Files are assigned protection levels for each of three groups of users: owner, users with a common project number, and all users. Each user group may be assigned a different access privilege. There are eight levels of protection, in each of the three user groups, that may be selected. The owner of a file may always change the file's protection.

File DAEMON is called on all access failures if the owner's protection is 4, 5, 6, or 7. An access failure occurs when an unauthorized user attempts to access a file. With a user-written program, the owner of a file may determine who attempted this unauthorized access.

Table 4–1 summarizes this file protection scheme of TOPS–10.

Table 4–1
File Protection Scheme

Protection Level	Access Code	Access Privileges
Greatest Protection	0	No access allowed
	1	Execute
	2	Read
	3	Allocate
	4	Deallocate
	5	Append
	6	Update
	7	Create
	10	Supersede
	11	Truncates
	12	Change attributes
	13	Delete
	14	Change name
Least Protection	15	Change protection

File Handling

To reference a file, the user does not need to know where the file is physically located in the system. A list, or directory, of all files for each user is contained in a User File Directory (UFD) or Sub-File Directory (SFD). These directories contain pointers to information concerning the file, its protection, its creation date, and information that is used to locate the file physically on the file structure. Each logical disk structure is complete; that is, all information necessary to uniquely locate files on that disk structure is contained within that structure. The operating system utilizes a user information table to determine which of the logical disk structures contain files for a given user. This SEARCH LIST may be changed by a user for himself, or by the operator, when the file structure configuration of the system changes, for example, a "private pack" is added to the system for a given user.

A named file is uniquely identified in the system by a file name and extension, an ordered list of directory names (UFDs and SFDs) that identify the owner of the

file, and a file structure name that identifies the group of disk units where the file is stored.

The file handler provides the interface between the user, operating system, and actual physical disk unit for storing and retrieving files on the disk.

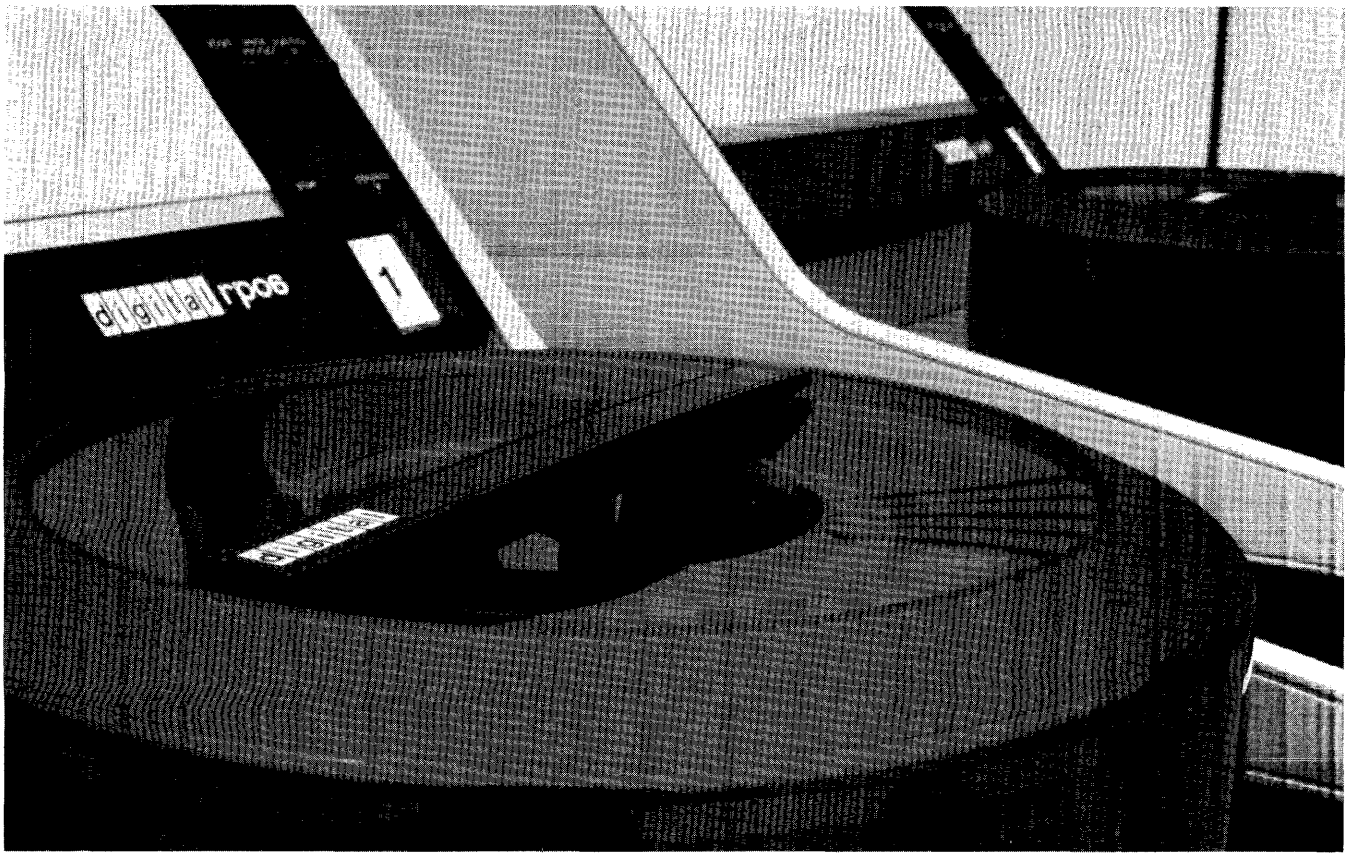
File Structures

Usually a complete disk system is composed of many disk units of the same or different types. Therefore, the disk system consists of one or more file structures. A file structure is a logical arrangement of files on one or more disk units of the same type. Because each file structure is self-contained, file structures can be added to or removed from the system during operation without reloading the system. File structures can also be moved from one DECsystem-10 to another DECsystem-10.

This method of file storage allows the user to designate which file system he wishes to use when storing files. Otherwise, the system refers to the Search List for the given user in the user administrative information tables.

Each file structure is self-contained and is the smallest section of file storage that can be removed from the system without disturbing other units in other file structures. All pointers to areas in a file structure are by way of logical block numbers rather than physical disk addresses; there are no pointers to areas in other file structures, thereby allowing the entire file structure to be removed.

A file structure contains two types of files: the data files that physically contain the stored data or programs; and the directory files that contain pointers to the data files. Included in these directory files are master file directories, user file directories, and sub-file directories. Each file structure has one master file directory (MFD). The MFD is the master list of all users of the file structure. The entries contained in the MFD are the names of all the user file directories on the file structure. Each user with files on the file structure has a user file directory that contains the names of all his files on that file structure; therefore, there are typically many user file directories on each file structure. The user can create another type of directory file: a sub-file directory (SFD). The SFD is similar to other types of directory files because it contains the names of all files within the sub-directory. This third level of directory allows groups of files belonging to the same user to be separate from each other. This is useful when organizing a large number of files according to function. In addition, SFDs allow nonconflicting, simultaneous runs of the same program using the same file names.



As long as the files are in different SFDs, they are unique. SFDs exist as files pointed to by the user file directory, and can be nested to the depth specified by the installation at system installation time.

File Protection

All disk files are composed of two parts: data and information used to retrieve data. The retrieval part of the file contains the pointers to the entire file, and is identically stored in two distinct locations on the structure and accessed separately from the data. System reliability is increased with this method because the probability of destroying the retrieval information is reduced; system performance is improved because the number of positionings needed for random-access methods is reduced. The storing and retrieval information is the same for both sequential and random accessed files. Thus, a file can be created sequentially and later read randomly, or vice-versa, without any data conversion.

One section of the retrieval information is used to specify the protection associated with the file. This protection is necessary because disk storage is shared among all users, each of whom may desire to share files with, or prevent files from being written, read, or deleted by other users. As discussed above, these protection codes are assigned by the user when the file is created and designate the level of user access to the file.

Disk Quotas

Disk space quotas are associated with each user on each file structure in order to control the amount of information that can be stored in the user file directory. When the user gains access to the system, he automatically begins using his logged-in quota. This quota is not a guaranteed amount of space, and the user must compete with other users for it.

When the user logs out of the system, he must be within his logged-out quota. This quota is the amount of disk storage space that the user is allowed to maintain when he is not using the system. The quota is enforced by a system program that is used in logging-out of the system. Quotas are determined by the individual installation, set up by the system manager, and are therefore, used to ration disk resources in a predetermined manner.

File Operations

Writing files on the disk can be defined by one of five methods: creating, superseding, updating, appending, and simultaneous updating. The user is creating a file if no other file of the same name exists in the user's file directory on the indicated file structure. If another file with the same name exists in the directory, the user is superseding , or replacing, the old file with the new file. Other users sharing the old file at the time it is being superseded continue to use the old file. They are not affected until they finish using the file and then try to access it again. At that time, they automatically read the new file. Thus, it is possible to have one user superseding the file and multiple users reading the file. However, if a second user attempts to supersede the file, the operating system prevents such an update. When a user updates a file, he modifies selected parts of the file without creating an entirely new version of the file. This method eliminates the need to recopy a file when making only a small number of changes. If other users try to access a file when it is being updated, they receive an error message issued by the system. Many users can update the same file through the use of a simultaneous update feature in TOPS–10. This feature allows users to work with the same data base concurrently. An Enqueue/Dequeue (ENQ/DEQ) facility is also offered to allow record-level lock out which eliminates race conditions, where multiple users are attempting to read and/or write the same record.

Disk Storage Management

File storage is dynamically allocated by the file handler during program operations, so the user does not need to give initial estimates of file length or the number of files to be written. Files can be any length, and users may have as many files as desired, as long as disk space is available and the user has not exceeded his logged-in quota. This feature is extremely useful during program development or debugging when the final size of the file is still unknown. However, for efficient random access, a user can reserve a contiguous area on the disk if he desires. When he has completed processing, he can keep his preallocated file space for future use or return it so that other users can have access to it, within the constraints of the user logged-out disk quota.

THE SCHEDULER

The TOPS–10 operating system is a multiprogramming system; that is, it allows several user jobs to reside in memory simultaneously and to operate concurrently. The scheduler determines which job should run at a given time. In addition to the multiprogramming implementation of TOPS–10, a swapping technique is used whereby active jobs can exist on an secondary storage device, such as disk, as well as in memory. The scheduler must therefore determine not only what job is to be run next, but also when a job is to be swapped out to secondary storage and later brought back into memory.

All jobs are retained in ordered groupings called queues. These queues have various priorities that reflect the status of each job at a given moment. The

queue in which a job is placed depends upon the system resource for which it is contending for with other active jobs. Because a job can wait for only one resource at a time, the job can only be in one queue at a time. Several of the possible queues in the system are:

- Run queues — for jobs waiting for, or jobs in, execution
- I/O wait queues — for jobs waiting for data transfers to be completed
- Resource wait queues — for jobs waiting for some shared system resource
- Null queue — for all job numbers that are not currently being used by an active user

The job's position within certain queues determines the priority of the job with respect to other jobs in the same queue. For example, if a job is first in the queue for a sharable device, it has the highest priority for the device when the device becomes available. However, if a job is in an I/O wait queue, it remains in the queue until the I/O is completed. Therefore, in an I/O wait queue, the job's position has no significance. The status of a job is changed each time it is placed into a different queue.

The scheduling of jobs into different run queues is governed by the system clock. This clock divides the CPU time into fixed-time quantas. When the clock ticks, the scheduler decides which job should run during the next clock cycle. Each job, when it is assigned to run, is given a time slice. When the time slice expires for a job, the clock triggers the scheduler and scheduling is again performed. The job whose time slice just expired is moved into another, perhaps lower priority, run queue, and the scheduler selects another job to run in the next time slice. If the currently running job is a null job (a null job runs when no user/system job is runnable), and a higher priority job (any job), becomes ready to run before the clock ticks, The higher priority job is run immediately. If the current running job is not a null job and a high priority (HPQ real-time) job becomes runnable, the HPQ is run at the next clock tick. Finally, if a job just becoming runnable is not an HPQ job, but is of higher priority than the current job, the new job preempts the current job when the current job's time slice has expired.

Scheduling may be forced before the clock ticks if the currently running job reaches a point at which it cannot immediately continue. When an operating system routine discovers that it cannot complete a function requested by the job, such as waiting for an I/O transfer to complete or discovering that a job needs a system resource which it currently does not have, the scheduler is called so that another job can be selected. The job that was stopped is then requeued, and is scheduled to run when the function it requested can be completed. For example, when the currently running job begins reading from a magnetic tape, it is placed into the I/O wait queue and the input is begun. A second job is scheduled to run while the input of the first job proceeds. If the second job then decides to access a magnetic tape, it is not placed into the I/O wait queue because the magnetic tape control is busy, but is placed in the queue for jobs waiting to access the magnetic tape. A third job is set to run. The input operation of the first job finishes, freeing the magnetic tape control for the second job. The I/O operation of the second job is initiated, and the job is transferred from the device wait queue to the I/O wait queue. The first job is then transferred from the I/O wait queue to a run queue. This permits the first job to preempt the running of the third job. When the time slice of the first job becomes zero, it is moved into the second run queue. The third job runs again until the second job completes its I/O operation or until the job's time slice for the second job expires.

This multiple queue design and implementation allows the operating system, through the scheduler module, to overlap computation and data transmission. In unbuffered data modes, the user supplies an address of a command list containing pointers to locations in his area to and from which data is to be transferred. When the transfer is initiated, the job is scheduled into an I/O wait queue where it remains until the device signals the scheduler that the entire transfer has been completed.

In buffered modes, each buffer contains information to prevent the user and the device from using the same buffer at the same time. If a user requires a buffer currently being used by the device, the user's job is scheduled into an I/O wait queue. When the device finishes using the buffer, the device calls the scheduler to reactivate the job.

TOPS–10 provides the ability to tune the scheduler to specific system loads and user environments. The System Manager, through privilege granted only to certain users, can set aside percentages of the CPU to given classes of timesharing and batch users. These percentages can be dynamically changed while the system is running and need not be defined at system installation time. This feature facilitates optimal system usage for a wide variety of system loads.

SMP Scheduling

Scheduling in a multiprocessor system provides more flexibility than a single-CPU system. TOPS–10 gives designated jobs and interactive work higher priority than "normal priority" and compute-bound work. TOPS–10 attempts to run higher priority and interac-

tive jobs when they request the CPU; other jobs are run in the "background". Optionally, the system manager can partition background jobs into classes, and allocate percentages of CPU time to individual classes.

With SMP, one CPU processes the high-priority and interactive jobs. The other CPU looks for normal priority work first, and processes such jobs as long as they are available; designated high-priority jobs and interactive jobs are serviced by this CPU only if there is no other work to do. This "asymmetric scheduling" has the basic effect that one CPU works on interactive jobs, while the other CPU runs compute-bound jobs.

An important aspect of scheduling in a multiprocessor environment is inter-CPU interference. For example, if both CPUs enter the scheduling routines simultaneously, they can compete for accesses to instructions and data, and , even more significantly, cause each other to wait for various interlocks (such as the one to prevent both CPUs from selecting the same job to run). SMP eliminates this problem by skewing the clocks on all CPUs to ensure that their clock interrupts occur at different times. Each CPU receives the same frequency of timer interrupts, but none occur at the same time as interrupts on the other CPU. Thus, CPU clocks in SMP are intentionally skewed to prevent periodic simultaneous scheduling and servicing overhead.

Under SMP scheduling, the queued I/O protocol ensures that I/O requests are handled properly, regardless of which CPU executes a job or where the job's files and devices are physically located in the system.

THE SWAPPER

The swapper is responsible for keeping the job most likely to run in main memory. It determines if a job should be in memory by scanning the queue in which the job may resides at a given moment. If the swapper determines that a job should be brought into memory, it may have to take another job already in memory and transfer it to secondary storage. Therefore, the swapper is not only responsible for bringing jobs into memory, but is also responsible for selecting the job that is to be swapped out to secondary storage.

A job is swapped out to secondary storage for one of two reasons:

- A job that is more eligible to run needs to be swapped in, and there is not enough room in memory for both jobs and the job being swapped to secondary storage is blocked.
- The job needs to expand its memory size and there is not enough room in memory to do so, because other jobs are currently resident.

If the latter case is true, the job must be swapped out to secondary storage and then later swapped in to memory with a new allocation of memory.

The swapper periodically checks to see if any jobs should be swapped into memory. If there is no such job, then the swapper checks to see if a job is requesting more memory. If there is no job wishing to expand its size, then the swapper does nothing further and relinquishes control of the processor until the next clock interrupt.

THE UUO HANDLER

The UUO handler is responsible for accepting requests for services available from the operating system. These requests are made by the user program with special instructions known as programmed operators or UUOs. When a programmed operator is encountered in the user program, the hardware does not execute the instruction, but rather "calls" the operating system. The operating system then examines the instruction, determines the function that the user program requires, and either initiates the operation (such as an I/O operation), or, if the desired resource is in use, places the request in the appropriate queue. The operating system then calls the scheduler, which in turn selects another job to run.

The services obtainable by the user program through UUOs include:

- Communicating with I/O devices on the system, including connecting and responding to any special real-time devices
- Receiving or changing information concerning either the system as a whole or the individual program
- Altering the operation of the system, such as controlling execution of a job, by trapping or suspending (for example, debugging purposes) or controlling memory by locking a job in memory (if the user has the appropriate locking privileges)
- Communicating, and transferring control between user programs

The UUO handler is the only means by which a user program can return control to the operating system in order to have a service performed. After the hardware has given control to the operating system and the function has been completed (often requiring a wait operation), control is returned to the user program at the instruction immediately following the UUO instruction, or some other location under certain error conditions.

In this way, the software supplements the hardware by providing services that are invoked through the execution of a single memory location, just as the hardware

services are invoked for a single instruction execution. This concept provides for efficient user programming and frees the user from any contention or complexity inherent in multitask, multiuser systems.

THE INPUT/OUTPUT ROUTINES

I/O programming under TOPS–10 is highly convenient for the user because all the burdensome details of programming are performed by the operating system. The user informs the operating system of I/O requirements by means of UUOs contained in the program. The input/output routines needed are called by the UUO handler.

Because the operating system channels communication between the user program and the device, the user does not need to know all the peculiarities of each device on the system. In fact,similar user programs can be written for all similar devices. Without returning an error message, the operating system ignores operations that are not pertinent to the device being used. Thus, a terminal and a disk file can be processed identically by the user program. In addition, user programs can be written to be independent of any particular device. The operating system allows the user program to specify a logical device name, which can be associated with any physical device at the time the program is executed. Because of this feature, a program that is coded to use a specific device need not be changed at all, if the device is unavailable. The device can be designated by a logical device name and assigned to an available physical device with one command, given at run time, to the operating system.

Data is transmitted between the device and the user program in one of two modes: unbuffered or buffered. With unbuffered data modes, the user in his program supplies the device with an address which is the beginning of a command list. Essentially, this command list contains pointers specifying areas in the user's allocated program space (which is located in memory) to or from which data is to be transferred. The user program then waits until the operating system signals that the entire command list has been processed. Therefore, during the data transfer, the user program is idle, waiting for the entire transfer to be completed.

Data transfers in buffered mode utilize a ring of buffers set up in the user's memory area. Buffered transfers allow the user program and the operating system's I/O routines to operate asynchronously. As the user program uses one buffer, the operating system processes another one by filling or emptying it as interrupts occur from a physical device. To prevent the user and the operating system from using the same buffer at the same time, each buffer has a use bit that designates who is using the buffer. Buffered data transfers are more efficient because the user program and the operating system can be working together in processing the data.

Several steps must be followed by the user program in order for TOPS–10 to have the information needed to control data transfers. Each step is indicated to the operating system with one programmed operator (UUO). In the first step, the specific device to be used in the data transfer must be selected and linked to the user program with one of the software I/O channels available to the user's job (the OPEN or INITialize programmed operators). This device remains associated with the software I/O channel until it is disassociated from it (via a programmed operator) or a second device is associated with the same channel. In addition to specifying the I/O channel and the device name, the user program can supply the initial file status, which includes the type of data transfer to be used with the device (such as ASCII, binary,etc.), and the location of the headers to be used in buffered data transfers. The operating system stores information in these headers when the user program executes programmmed operators. The user program obtains from these headers all the information needed to fill empty buffers.

Another set of programmed operators (INBUF and OUTBUF) establishes the actual buffers to be used for input and output. This procedure is not necessary if the user is satisfied to accept the default buffers automatically set up for him by the operating system.

The next step is to select the file that the user program is using when reading or writing data. This group of programmed operators (LOOKUP and ENTER) is not required for the devices that are not file-structured (such as card reader, magnetic tape, line printer, etc.). However, if used, they are ignored, thus allowing file-structured devices to be substituted for nonfile-structured devices without the user rewriting the program.

The third step is to perform the data transmission between the user program and the file (IN, INPUT, OUT, and OUTPUT). When the data has been transmitted to either the user program on input or the file on output, the file must be closed (CLOSE, "fourth step") and the device released from the channel (RELEASE, "fifth step"). The FILOP.UUO combines OPEN, INBUF, LOOKUP, and ENTER into one operation for efficiency in programming. This same sequence of programmed operators is performed for all devices; therefore, the I/O system is truly device-independent, because the user program does not have to be changed every time a different device is used.

In addition to reading from or writing data to the standard I/O devices, provisions are included in the operating system for using the terminal for I/O during execution of the user program. This capability is also

obtained through programmed operators. As the user program is running, it can pause to accept input from or to type output to the terminal. The operating system does all buffering for the user, thus saving programming time. This method of terminal I/O provides the user with a convenient way of interacting with a running program.

MEMORY MANAGEMENT

TOPS–10 is a multiprogramming operating system, that is, it allows multiple independent user programs to reside simultaneously in memory and run concurrently. This technique of sharing memory and processor time enhances the efficient operation of the system, for example, by switching the processor from one program that is temporarily stopped because of I/O activity to a program that is executable. When memory and the processor are shared in this manner, each user's program has a memory area distinct from the area of other users. Any attempt to read or change information outside of the area a user can access, stops the program immediately and notifies the user. This protection is controlled by the TOPS–10 operating system and the hardware on which the operating system is running.

Because available memory can contain only a limited number of programs at any one time, the system employs a secondary storage unit, a disk, allowing programs to be swapped to less expensive and higher capacity secondary storage devices. User programs exist on this secondary storage device and are moved into memory for execution, as described above in the paragraph on the Swapper. Because this swapping takes place directly between main memory and secondary storage, the CPU can be executing a user program in one part of memory while swapping of another user program is taking place in another part of memory. This independent, overlapped operation greatly improves system utilization by increasing the number of users that can be accommodated at the same time.

To further increase the utilization of memory, the operating system allows users to share the same copy of a program or data segment. This prevents the excessive memory usage that results when a program is duplicated for each of several users. A program that can be shared is called a reentrant program and is divided into two segments. One segment contains the code that is not modified during execution and can be used by any number of users. The other segment contains nonreentrant code and data. The operating system provides protection for shared segments to quarantee that they are not accidentlly modified. Available language compilers, which operate under TOPS–10, are themselves reentrant; and many generate reentrant code.

Another important feature of the reentrant implementation is that a sharable segment of code needs to be swapped out to secondary storage only once. Because this segment cannot be modified, a copy is available on the secondary storage device, and can be swapped into memory as required. When the swapper determines that a sharable segment must be swapped out to make room for another user program, the operating system need only note that the memory segments previously occupied by the reentrant segment are available.

To efficiently manage memory, main memory is logically divided into 512-word pages. All systems running TOPS–10 support hardware page tables. Hardware page tables allow programs to be noncontiguous in main memory. During a swap- in operation, the operating system places a given user's program throughout memory, and loads the appropriate hardware page table pointing to these physical memory locations. This eliminates the need for memory "shuffling" and increases efficiency.

Virtual Memory

The TOPS–10 virtual memory option permits a user program to execute with an address space greater than the physical memory allocated to that program during execution. User programs are swapped, as described in the section on the Swapper. However, the entire program may not necessarily be in main memory during execution. Programs are divided into pages, each of which is 512-words long. Some of these pages may remain on secondary storage while the program executes. When a virtual memory job attempts to access a page that is not in memory, a page fault occurs. The page fault handler determines which page or pages to remove from memory and which pages to bring in from secondary storage.

Unlike virtual memory implementation on other systems, TOPS–10 provides this feature as an option. The system manager grants the privilege for using virtual memory only to those users who truly need its capabilities. Those users who are granted the privilege of using virtual memory may elect to invoke the feature for only those programs that could not execute without the virtual memory capability.

The virtual memory users may elect to use the system page fault handler to decide which pages are brought into memory when apage fault occurs; or they may use a handler they write themselves that is more tailored to the particular application or program behavior. Finally, it is important to point out that only those users actually using the virtual memory feature are affected by any additional overhead associated with a demand-paging system. Nonvirtual memory users execute their jobs as they would in a nonvirtual memory

system with no discernable difference in performance, except possibly for the additional system overhead required for swapping and file channel access incited by the virtual job or jobs.

INTER–PROCESS COMMUNICATION FACILITY (IPCF)

IPCF provides the capability for independent jobs to communicate with one another. For example, if several programs or progamming systems are involved in processing or maintaining a data base, it is possible that one program might want to inform the others of any modifications it makes to the data. A job using the IPCF cannot make any changes to another job, so protection is in no way sacrificed when using the IPCF feature.

In order to use the IPCF, each participating job that wishes to receive communication from another job must request a unique process identifier (PID) from the system. The transmitting job then may send a "packet" of information to another job. In addition to the information, the system automatically provides a "return address" so that the receiving program can respond to the sender.

The operating system maintains a linear queue (the "mailbox") for each job using IPCF. The packet, or packets, are kept in the mailbox until the receiving job retrieves it. This queue is not created until a job sends an IPCF packet, and it does not occupy any space until such time. The maximum number of packets allowed in a queue at any one time is determined by a "receive" quota that may be set at each installation for each user by the System Manager. If no quota is set, the system default is used. The system default is five.

On systems with the virtual memory option, the packet could be an entire page. In this case, the operating system takes advantage of the page-mapping hardware of the CPU to transmit the page without actually copying the page. This procedure is called "page passing".

COMMUNICATION SOFTWARE

TOPS–10 provides a wide selection of communications capabilities to enhance or facilitate the user's computing needs. Communications functionality has always been an integral part of the TOPS–10 operating system. With its long heritage of interactive, multimode computing, user access to the features and functionality of the operating system has always been an important design consideration of any implementation of the operating system.

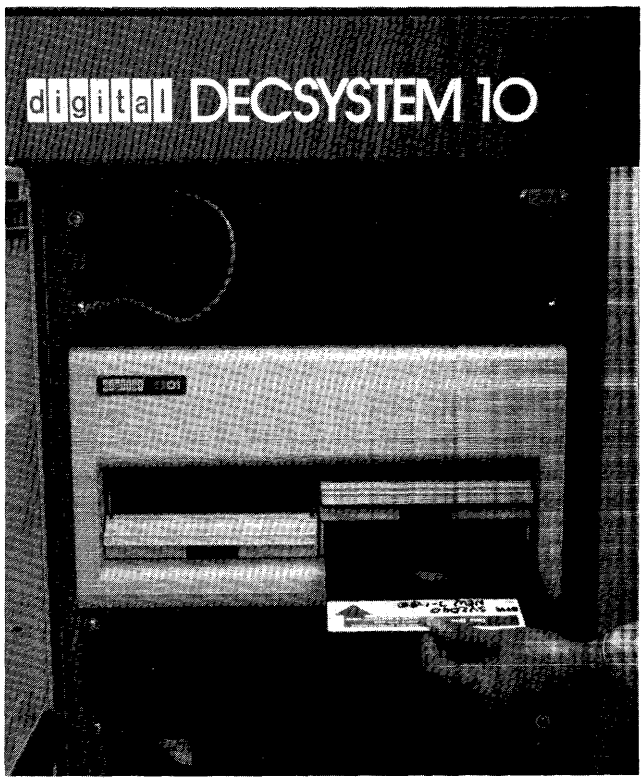
Within the multitask environment of TOPS–10, functionality exists for:

- Asynchronous communication — the typical method for interactive timesharing, transaction processing, program development and debugging, and data entry
- Synchronous communications — for connection of remote batch stations, remote terminal comcentration, and computer-to-computer links

The TOPS–10 operating system handles the communications housekeeping, and all communication functions are fully supported within the operating system. Appropriate synchronous line protocols for interfacing to non-DIGITAL products are supported.

Networks with simple or complex topologies can be configured utilizing various families of front-end processors, communications processors, remote batch stations, remote terminal concentrators, and a combination of remote batch station and remote terminal concentrator products. These products feature functionality throughout a wide spectrum of communication needs.

Additional details concerning TOPS–10 based communication products can be found in Section 10 of this document.



CONSOLE FRONT–END PROCESSOR AND SOFTWARE

On the DECsystem-10, the console front-end processor and software reduce the central processor's participation in I/O operations and serve as a powerful diagnostic/maintenance tool for service personnel.

Specifically, the console front-end processor is responsible for providing:

- Console functions
- Interface for command terminals
- Interface for unit record peripherals
- Diagnostic and maintenance functions

Console Functions

Two major functions of the front-end are system initialization and communication between the system and operator. The user need only push a button and type in data to initiate the following automatic program sequence. The console front-end processor loads and verifies the microcode, configures and interleaves memory, and loads and starts execution of the central processor bootstrap program from a device specified by the user.

The user can request a memory configuration listing that indicates which memory controller is online, what the highest memory address configured is, and how the memory is interleaved.

All normal console capabilities, such as the ability to examine and change registers, start, stop, reset, and load, are provided by the console front-end processor. An operator command language is provided with the system.

Command Terminal Functions

The console front-end processor serves as an intelligent data link and buffer for the interactive terminals, relieving the central processor of this overhead. The console front-end processor buffers the characters received and interrupts the central processor, either upon receipt of a clock signal or in the event it has a group of characters to send. The central processor also sends groups of characters for terminal output to the console front-end processor, further enhancing efficiency. Programmable terminal speed-setting capability is provided, enabling terminal speeds to be changed dynamically. On dialup terminal lines, the monitor will automatically select the appropriate rate for 110, 150, 300, or 1,200 baud lines when the user types RETURN or CTRL C. Thus, line speeds need not be associated with phone numbers.

Peripheral Interface

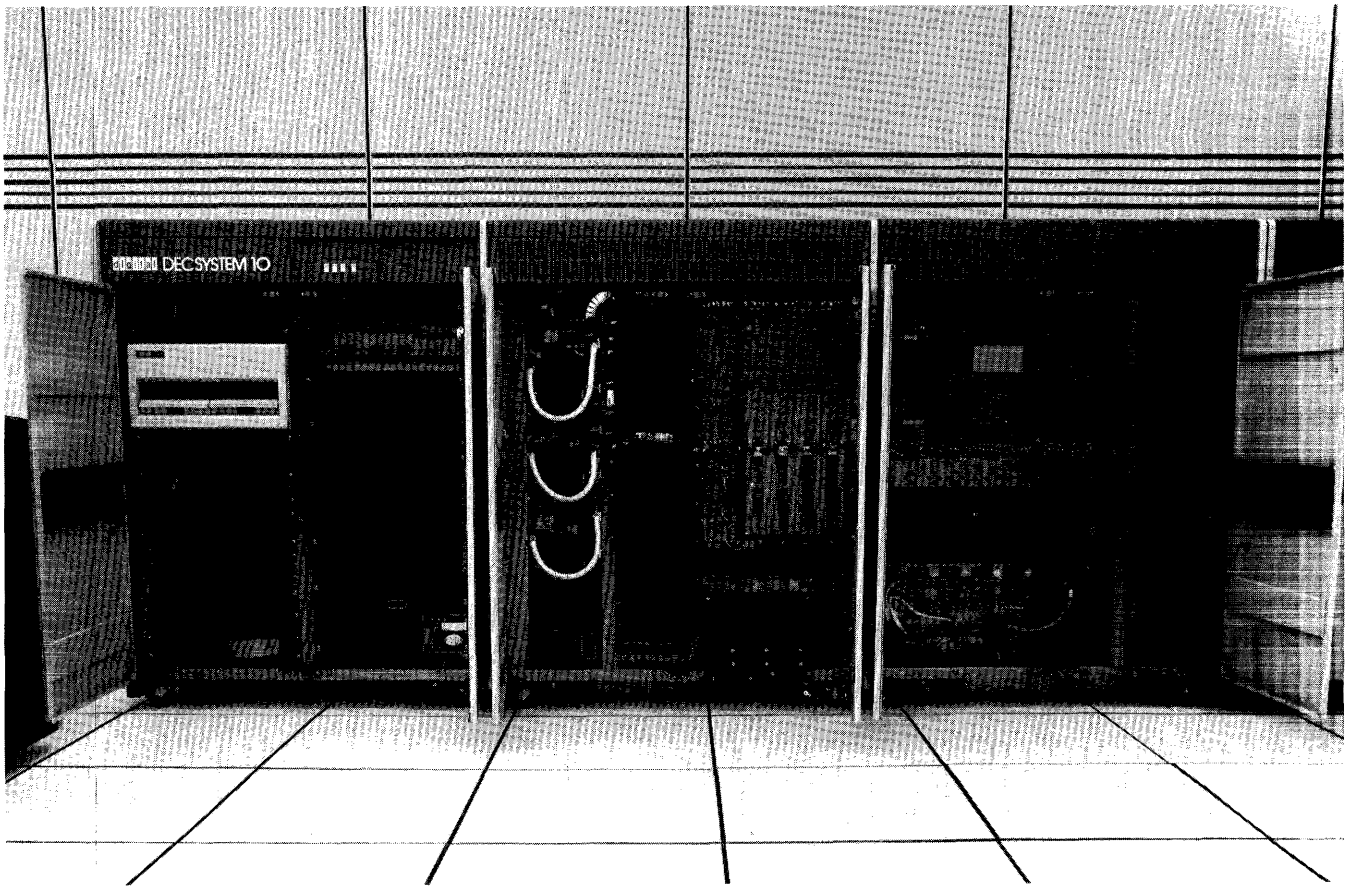
The unit record-spooling programs in TOPS–10 communicate with the console front-end processor, using the same buffering mechanism described for the command terminal functions. Both the central processor and the console front-end processor maintain buffers for data. They interrupt only once per buffer transmission, increasing the amount of useful work each processor can do.

Diagnostic Maintenance Functions

Remote diagnosis capability reduces mean time to repair (MTTR) and increases system availability to the user. It also allows DIGITAL's service engineers to determine the causes of most system failures before leaving the local office. By running diagnostic tests using telephone dialup facilities, the Field Service engineer can give the customer better service because it is easier to select which spares and test equipment should be taken on the call. Certain tests can be run during general timesharing. Even if the main CPU is inoperative, the console front-end processor can access all buses and major registers.

Total system security is maintained because the onsite system operator must activate the communications link, and only the onsite personnel know the specific password to the system each time remote diagnosis is employed. Time limits for system access can also be imposed, and the remote link cannot operate at a higher privilege level than that specified by the local console terminal. All input and output to the diagnostic link is copied to the local terminal, giving onsite personnel a record of all steps taken.

5
The KL10 Central Processor



The KL10 central processing unit is the heart of DIGITAL's large scale computer systems and has provided the basis for the popular and successful DECsystem-10 computer family.

The KL10–E version of the KL10 is the central processing unit of the DECsystem-1091. The KL10–D version of the KL10 is the central processing unit of the DECsystem-1090.

Important features of the KL10 central processing unit are:

- 398 logically-grouped microprogrammed instructions, including byte and string manipulation, double-precision fixed and floating-point, and character editing and conversion
- An architecture that allows modular expansion without component replacement or complex reconfiguration
- An integrated PDP–11 front-end processor for diagnostic functions, console functions, and handling low-speed peripheral devices
- Extensive error-detecting circuitry for maintainability and availability
- Integrated high-speed data channels/controllers for disks and tapes for improved reliability and price/performance
- State-of-the-art memory management hardware that allows virtual memory operation
- Four-word memory fetch capability for increased bandwidth and performance
- High-speed cache memory with state-of-the-art design. Typical on-line programming results in a 90% cache "hit" rate for efficiency and high performance
- A full line of high-efficiency, high performance UNIBUS and MASSBUS peripheral devices to meet individual user needs
- High-speed ECL logic implementation
- Eight sets of general purpose registers which can be used as accumulators, index registers, or the first 16 locations of main memory, and which have an access time of 160 nanoseconds

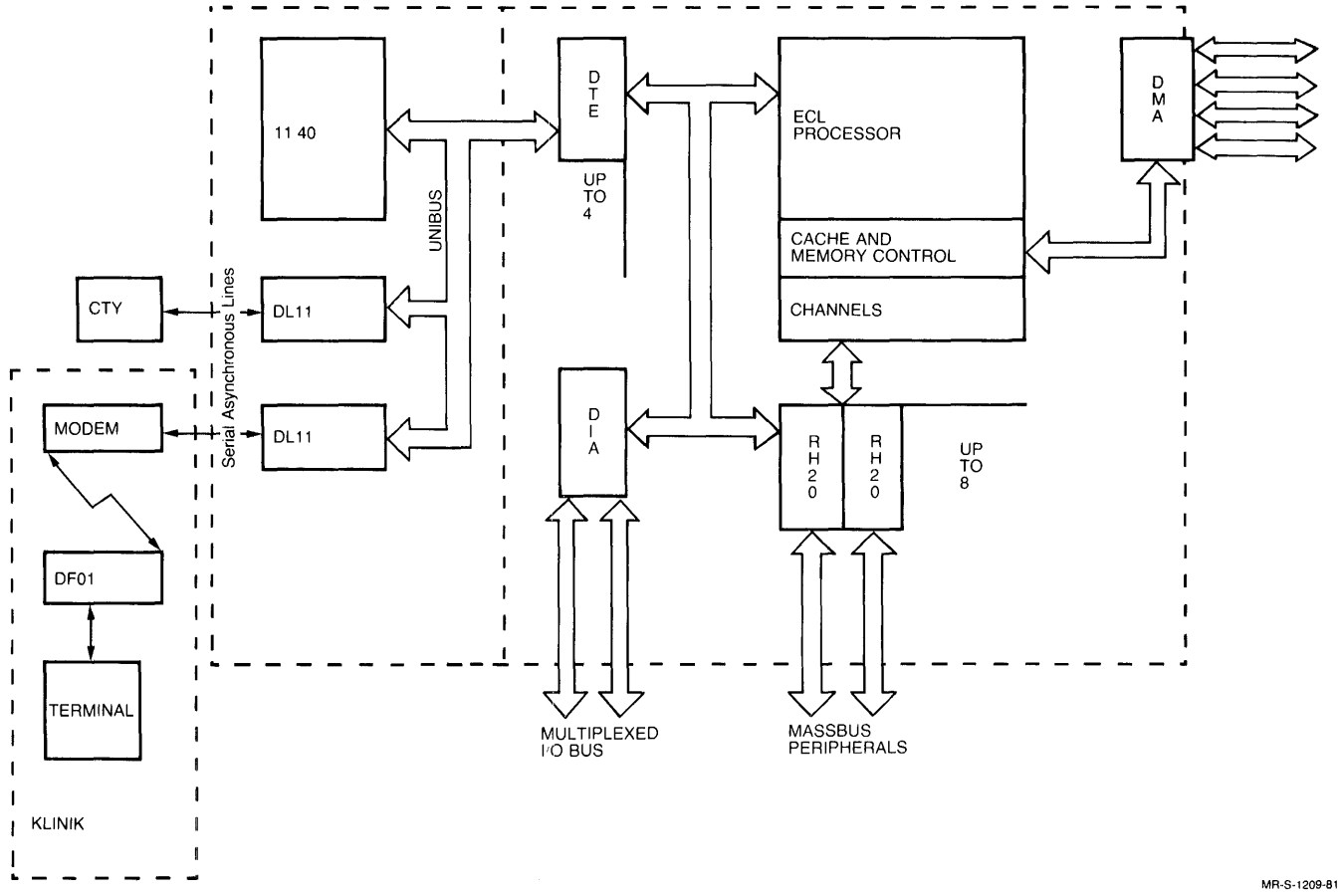


Figure 5–1
KL10–D Central Processing Unit

SYSTEM ARCHITECTURE

The internal architecture of the KL10 central processing unit may be considered a collection of "special purpose" processors connected through high-speed internal buses. Four major subsystems constitute the DECsystem-10 architecture for purposes of this discussion:

- The Execution Box (E–Box) and associated buses, which execute machine instructions and provide control for other subsystems
- The main memory subsystem, which consists of a memory control unit (M–Box), associated buses, and internal memory, and/or MOS memory or external core memory, depending upon the physical main memory configuration
- The front-end diagnostic and console PDP–11 subsystem, which also supports UNIBUS unit-record equipment
- The I/O subsystem, which consists of integrated channel/controllers, a multiplexed I/O bus (on some models), and PDP–11-based communications subsystems

Figure 5–1 illustrates a fully configured KL10–D based DECsystem-10. Key elements of the packaging and implementation of the KL10–D processor are External Memory Buses terminating in the DMA interface, a Multiplexed I/O Bus terminating in the DIA interface, MASSBUS for high-speed peripherals (disks, tapes) terminating in RH20 integrated channels and controllers, and a UNIBUS for low-speed peripherals (unit-record and PDP–11-based communications) terminating in the DTE interface. (The DTE is discussed later in this section.)

Figure 5–2 illustrates a fully configured KL10–E-based DECsystem-10. Key elements of this packaging and implementation of the processor are internal memory, an optional multiplexed I/O bus terminating in the DIB interface, a MASSBUS for high-speed peripherals (disks, tapes, etc.) terminating in RH20 integrated channels and controllers, and a UNIBUS for low-speed peripherals (unit-record and PDP–11-based communications interfaces) terminating in the DTE interface.

These two figures point out the important differences in the two versions of the KL10 CPU. The KL10–E supports internal memory for TOPS–10 and may include a DIB I/O bus interface (if required for certain peripherals); the I/O bus is standardly configured in the KL10–D.

Figure 5–3 further defines the KL10–D and KL10–E from a packaging standpoint. The KL10–D is packaged in 72″ cabinets; the KL10–E utilizes DIGITAL Corporate 60″ "high-boy" cabinets.

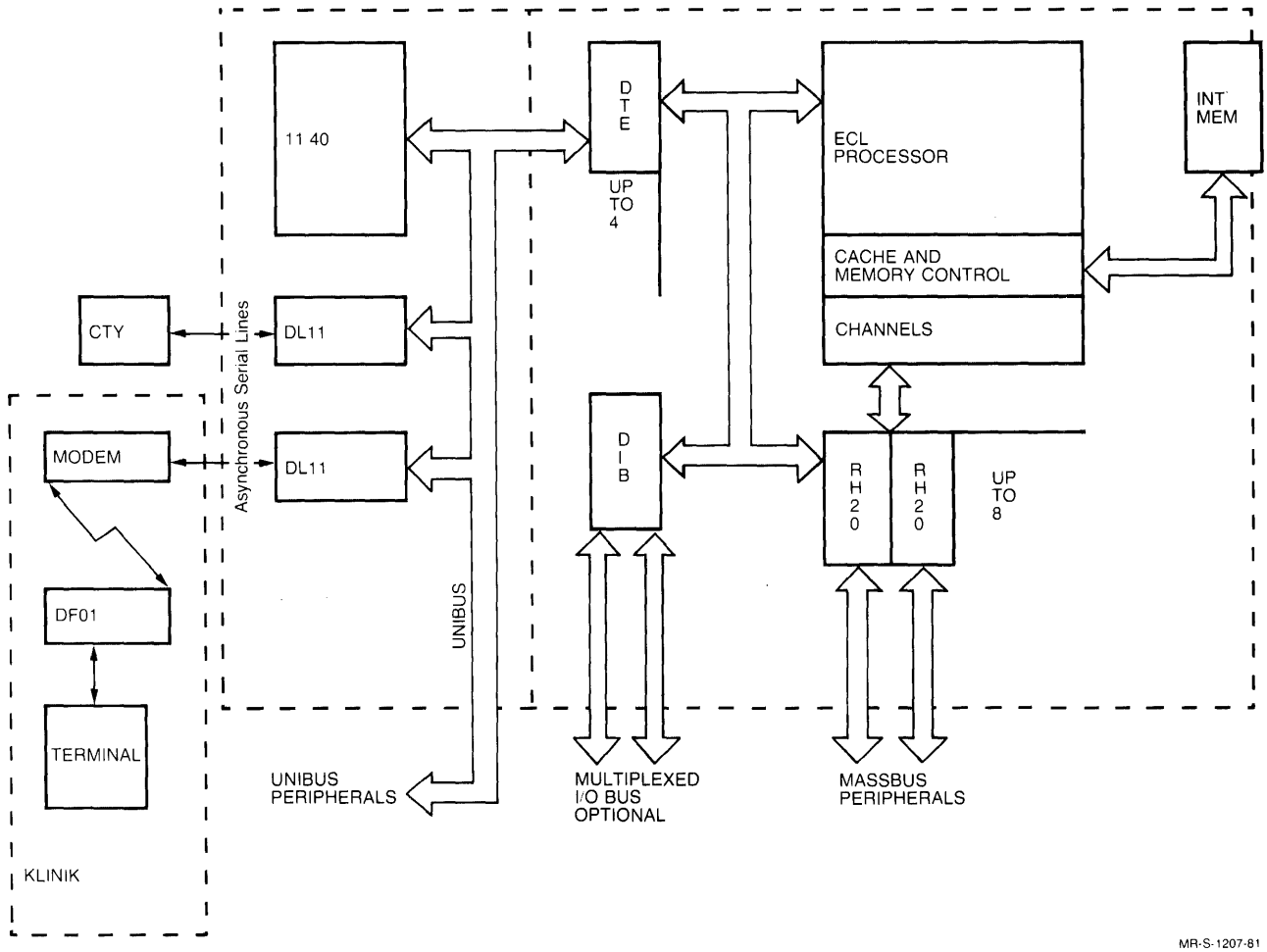
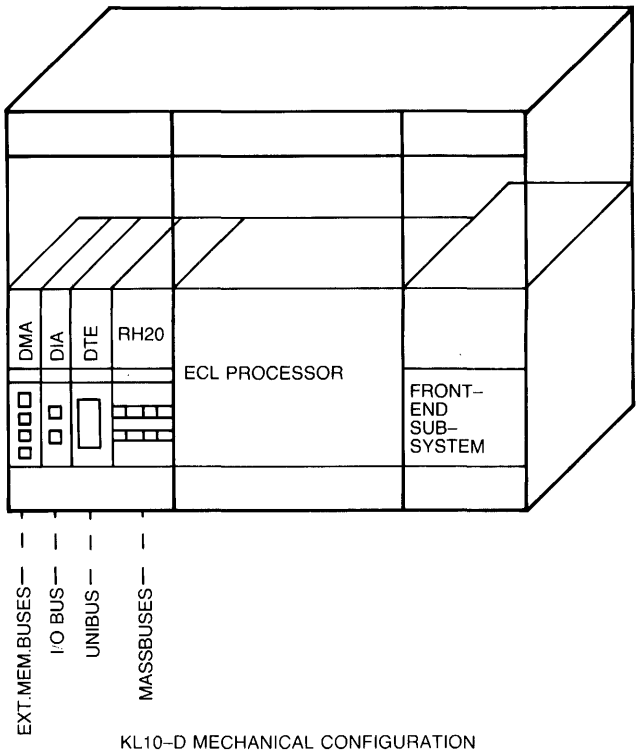
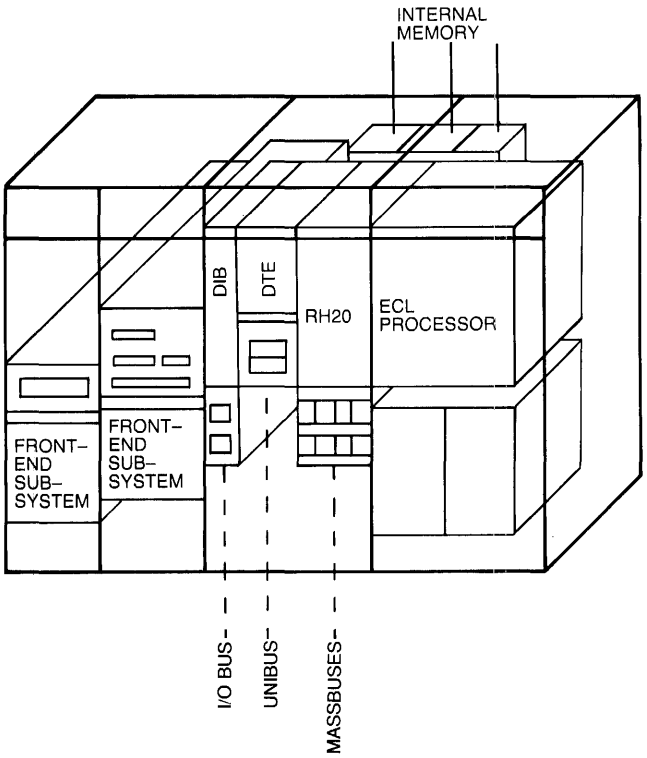


Figure 5–2
KL10–E Central Processor Unit



KL10–D MECHANICAL CONFIGURATION



KL10–E MECHANICAL CONFIGURATION

MR-S-1231-81

Figure 5–3
KL10–D/E Mechanical Configuration

Execution Box (E–Box)

The execution box (E–Box) is the element of the processor that actually executes instructions and performs various control functions for overall CPU operation. The KL10 features 398 microprogrammed instructions. Each instruction is a series of microinstructions that performs various logical functions like processor state control, data path control, and the actual implementation of each instruction. This is important to the overall flexibility of the KL10 central processor. Through the use of microcode (as opposed to "hard-wired" instruction implementation), a machine instruction executes one or more microstore instructions. To execute a given machine instruction, the microinstruction sequence associated with that instruction performs the data movement, operations on data, or control steps that are needed to execute instructions. The microprogrammed KL10 central processor offers several important advantages:

- The CPU executes TOPS–10 paging (memory management) and monitor call interface.
- Engineering changes or central processor improvements can often be implemented by microstore (firmware) changes, rather than by wiring changes, since the microstore is loaded by the PDP–11 front end during system boot program.
- Cache and virtual memory operations are enhanced by using microprogramming.
- The packaging technology of microcoded hardware permits smaller size and greater flexibility in instructions.
- Architectural extensions can often be made using microprogramming, which eliminates the need for extensive engineering change orders (as, for example, in the support of new peripheral subsystems).

The microstore is actually implemented with 2,048 75-bit words. The E–Box also has a 512-word instruction dispatch table that rapidly decodes instructions and dispatches operations to the appropriate microstore sequence to implement the instruction. Besides performing the basic functions of decoding and implementing control store steps for the implementation of each instruction in the KL10 set, the microcontroller executes the more fundamental operations of sequencing the program and processing interrupts.

The E–Box also contains eight sets of 16 general purpose registers, four timing and accounting meters, and an interface to the low-speed I/O devices (DTE for UNIBUS equipment and DIA/DIB for multiplexed I/O bus devices). The interface to the low-speed control devices is controlled by the E–Box's microcode and performs the required data and control transfers by stealing microcode cycles between the execution of CPU instructions.

Instruction Set

The KL10 has 398 microprogrammed instructions, (an extremely large repertoire), which provides the flexibility required for specialized computing problems. Since the set provides so wide a selection of instructions from which to choose, few are typically required to perform a given function. Programs can therefore be shorter than those of other computers. The large instruction set also simplifies the monitor (operating system), language processors, and utility programs.

In addition to the 398 instructions, the KL10 provides 64 programmable operators of which 33 trap to the monitor and 31 trap to the user's call area. The remaining instruction codes are unimplemented and reserve for future expansion. An attempt to execute one of the unimplemented instructions results in a trap to the monitor.

Despite its size, the instruction set is easy to learn. The set is logically grouped into families of instructions, and the mnemonic codes are constructed modularly. All instructions are capable of directly addressing a full 256K (K 1,024) 36-bit words of memory without resorting to base registers, displacement addressing, or indirect addressing. Instructions can, however, use indirect addressing with indexing to any level. Most instruction classes, including floating point, allow immediate mode data. The result of this is effective address calculation used directly as an operand to save storage and speed execution.

The processor mode concept has been a keystone in DIGITAL's timesharing or multitasking systems for over a decade, and it is central to the KL10 implementation. Instructions are executed in either user mode or executive mode. In a multitask environment, a user must not execute instructions that would jeopardize other users. On the KL10, attempted execution of such an instruction simply traps to the operating system, which analyzes the instruction and takes appropriate action.

In executive mode operation, any implemented instruction is legal. The monitor operating in executive mode is able to control all system resources and the state of the processor. Executive and user modes are further divided into two submodes each. Table 5–1 defines the processor mode implementation.

Table 5–1
Processor Modes

User Mode
Public Submode
• User programs
• 256K word address
• All instructions permitted unless they compromise integrity of the system or other users
• Can transfer to concealed submode only at entry points
Concealed Submode
• Proprietary programs
• Can READ, WRITE, EXECUTE, or TRANSFER to any location labeled public
Executive Mode (monitor)
Supervisor Submode
• General management of system
• Those functions that affect only one user at a time
• Executes in virtual address space labeled public
Kernel Submode
• I/O for system
• Those functions that affect all users

Instruction Format

In all but input/output instructions, the nine high-order bits (0–8) specify the operation; and bits 9–12 usually address an accumulator (but are sometimes used for special control purposes, such as addressing flags). The rest of the instruction word always supplies information for calculating the effective address, which is used for immediate mode data, or is the actual address used to fetch the operand or alter program flow. The effective address is referred to throughout this document as the user (or operating system) virtual address. Memory-mapping and/or paging hardware can "map" this effective address into a very different "physical" address within main memory. But the programmer need only be concerned with an effective user address space that can be up to 256K words. Bit 13 specifies the type of addressing (direct or indirect). Bits 14–17 specify an index register for use in address modification (zero indicates no indexing), and the remaining eighteen bits (18–35) contain a memory address which is used as the basis for effective address calculation or as an actual operand in immediate mode format.

All input/output instructions are designated by "ones" in bits 0–2. Bits 3–9 address the device to be used, and bits 10–12 specify the operation. Bits 13–35 are the same as non-I/O instructions. They are used in calculating an effective address for the data to be transferred.

Figure 5–4 illustrates the two instruction formats for the KL10.

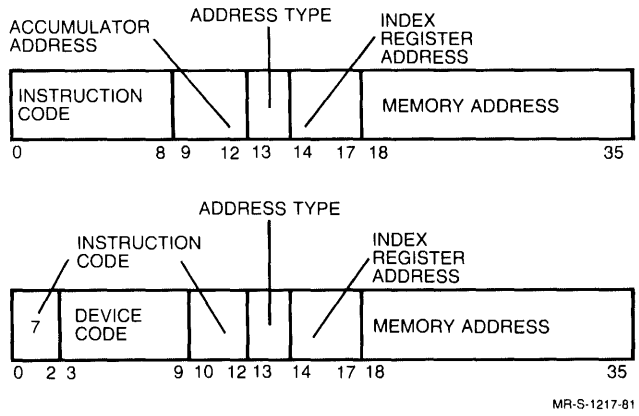


Figure 5–4
Instruction Formats

Half-Word Data Transmission

Half-word data transmission instructions move a half word and can modify the contents of the other half of the destination location. The 16 instructions differ in the direction they move the chosen half word and in the way they modify the other half of the destination location.

Full-word Data Transmission

Full-word data transmission instructions move one or more full words of data from one place to another. The instructions can perform minor arithmetic operations, such as calculating the negative or the magnitude of the word being processed.

Byte Manipulation

Five byte manipulation instructions pack or unpack bytes of any length anywhere within a word.

Logic Instructions

Logic instructions shift, rotate, and execute the 16 Boolean operations on two variables.

Fixed-Point Arithmetic

The KL10 is a 2s' complement machine in which zero is represented by a word containing all zeros. As in comparable implementations, the KL10 logic does not keep track of a binary point. The programmer must adopt a point convention and shift the magnitude of the result to conform to the convention used. Two common conventions are to regard a number as an integer (binary point to the right) or as a proper fraction (binary point to the left). In these cases the range of numbers represented by a single KL10 word is -2^{35} to 2^{35-1} (binary point to the right) or -1 to $1-2^{35}$ (binary point to the left). Since multiplication and division use double-length numbers, there are special instructions with integral operands for these operations.

The format for double-length fixed-point numbers is an extension of the single-length format. The magnitude (or its 2s' complement) is the 70-bit string in bits 1–35 of the high- and low-order words. Bit 0 of the high-order word is the sign; bit 0 of the low-order word is ignored. The range for double-length integer and proper fractions is thus -2^{70} to 2^{70-1} (binary point assumed at the right) or -1 to $1-2^{70}$ (binary point assumed at the left).

Floating-Point Arithmetic

The KL10 processors have instructions to scale, negate, add, subtract, multiply, and divide in single- and double-precision floating-point format. In the single-precision floating-point format, one bit is used for the sign, eight bits for the exponent, and 27 bits for the fraction.

In double-precision floating-point format, one bit is used for the sign, eight bits for the exponent, and 62 bits for the fraction. This results in a precision in the fraction of 1 part in 4.6×10^{18} and an exponent of 2 in the range from -128 to $+127$.

Normalized single-precision floating-point numbers have a fraction that can range in magnitude from 1/2 to $1-(2^{-27})$. Increasing the length of a number to two words does not significantly change the range, but increases the precision. In any format the magnitude range of the normalized fraction is from 1/2 to 1 less the value of the least significant bit. In all formats the exponent range is from 2^{-128} to 2^{127}.

Fixed-Floating-Point Conversion

Special instructions convert between fixed- and floating-point formats. Two sets of conversion instructions are provided, one optimized for FORTRAN, the other for ALGOL.

Arithmetic Testing

An arithmetic testing instruction can jump or skip, depending upon the result of an arithmetic test that the instruction can first perform on a test word.

Logical Testing, Modification, and Skip

These instructions modify and/or test using a mask and/or skip *n* selected bits in an accumulator.

Program Control

Program control instructions include several types of jump instructions and subroutine calls including calls that save the return address on a pushdown stack.

Input/Output Operations

Input/Output (I/O) instructions govern all direct trans-

fers of status to and from the peripheral equipment. They also perform many operations within the processor. Block transfer instructions handle bulk data transfers to and from medium-speed I/O bus devices. High-speed devices transfer data directly to memory through internal channels.

Unimplemented User Operations (UUOs)
Many of the codes not assigned as specific instructions are executed as unimplemented user operations; the word given as an instruction is trapped and must be interpreted by a routine included for this purpose either by the programmer or by the monitor. UUOs reserved for the monitor are called monitor UUOs (MUUOs). User UUOs are called local UUOs (LUUOs). Instructions that are illegal in user mode also trap in the same manner as MUUOs.

Business Instruction Set
There are five classes of instructions in the Business Instruction Set in the KL10 central processor. Four of these are arithmetic instructions to add, subtract, multiply, and divide using double-precision fixed-point operands. The STRING instruction in the fifth class can perform nine separate string functions.

These functions include an edit capability, decimal-to-binary and binary-to-decimal conversion in both offset and translated modes, move string in both offset and translated modes, and compare-string in both offset and translated modes. Offset mode is byte modification by addition of the effective address of the string instruction. Besides providing the translation function, these instructions can control AC flags and can detect special characters in the source string.

The Business Instruction Set provides faster processing because there are special instructions for a wide variety of string operations. These instructions can be used on a variety of code types such as ASCII and EBCDIC. The Business Instruction Set exemplifies a microprogrammed processor's advantages in meeting special user needs.

Trap Handling
The execution of programmed trap instructions permits the KL10 to directly handle arithmetic overflows and underflows and pushdown list overflows.

Fast Register Blocks
Eight sets of 16 general purpose registers (accumulators) are provided in the KL10 architecture. They can be used as accumulators, index registers, or as the first 16 locations in main memory. Since register addressing is included in the basic instruction format, no special instructions are needed to access these registers. Different register blocks are used for the operating system and for individual users, eliminating the need for storing register contents when switching from user mode to executive mode.

Programmable Address Break
When a specified location is properly referenced, a programmable address break suspends a user program and traps to the operating system. This facility is particularly useful in program development and debugging.

Meters
Four meters built into the KL10 provide a number of timing and counting functions, including an interval timer, time base, accounting meter, and a performance analysis counter.

The meters are controlled by I/O instructions to internal devices. Many of the timing functions use a microsecond pulse source originating in the basic 30-megacycle machine clock. Designed with a tolerance of 0.05 percent, this pulse source results in a less than a five-second drift in 24 hours.

The Interval Timer provides a programmable source of interrupts with a one microsecond resolution. The Interval Timer used for real-time applications and for page management by the monitor. The Interval Timer is designed so that a real-time schedule with varying deadlines can be implemented.

The Time Base provides a 60-bit, one-microsecond resolution counter that can be used to generate a source of elapsed time. This 60-bit register provides more than 9,000 years maximum time before wrap-around.

The Accounting Meters provide an accurate measurement of the amount of processor resources used by a job.

The Performance Analysis Counter provides a tool for studying hardware and software performance. Through its ability to measure both the number and duration of certain events, it may be used to measure such events as cache hit rate,interrupts, and channel activity.

Priority Interrupt System
The KL10 has been designed and implemented with a powerful, flexible priority interrupt system. It permits overlapped, concurrent, asynchronous operation of the central processor, peripheral controllers and devices, and software service routines.

Devices are assigned under program control to one of seven priority levels through dynamic loading of a 3-bit register within the device. Each of the seven levels has a number of programmable sublevels that enable the operating system to change the priority level of

any device or disconnect the device from the system and later reinstate it at any, or the same, level. Similarly, the operating system can set, enable, or disable all (or any combination of) levels with a single instruction. The operating system can assign some or all devices to the same level.

The system can also generate interrupts through software, so real-time hardware can operate on a high-priority level while related computations, particularly if they are lengthy, can be performed on a lower level.

The program-assignable priority interrupt system provides much greater flexibility than permanently hardwired systems, which require a large number of levels, often operate with an extremely high overhead, and cannot change device priorities without system shutdown and rewiring.

In conjunction with the priority interrupt system, a set of instructions (BLOCK–IN/BLOCK–OUT) allow blocks of information to be transferred between a device and memory in a single operation. When an interrupt occurs, the KL10 will trap to a predetermined location. The BLOCK–IN/BLOCK–OUT instruction identifies the source of the interrupt, transfers a word in or out, updates word count and data address, and dismisses the interrupt. When the word count is zero, a different action occurs that allows the program to either process the block of information that has been transferred or to move on to other activities.

Multiplexed I/O Bus

A multiplexed I/O Bus, which provides a 36-bit full-word parallel path between memory and an I/O device for purposes of control or low-speed data transfer, is standard on the KL10–D and optional on the KL10–E. To initiate such data transmission, a control word is first transferred over the I/O buffer of the device controller. Then, on command, data words or data blocks are moved directly to or from memory using the BLOCK–IN or BLOCK–OUT instruction as explained above.

The I/O bus can also be used as a control and data path to and from a large number of low-speed I/O devices. Transfer is performed in 36-bit words in parallel. Thus, each data transmission instruction moves one word of data between memory and the buffer of the device controller. When block input or output instructions are used, entire blocks of data are moved to or from the device with a single instruction.

Memory Subsystem

Central to the KL10 architecture is the control all memory through a discrete memory controller (M–Box). The M–Box is responsible for all memory requests from the E–Box and from integrated channel/controllers for high-speed peripherals (disks and tapes). The M–Box does paging (address translation) for memory management and contains the cache memory for the CPUs.

Physical Memory

Through the DMA External Memory Controller, the KL10–D supports external memory modules as illustrated in Figure 5–5; the KL10–E supports internal MOS (Metal-Oxide-Semiconducter) memory as illustrated in Figure 5–6.

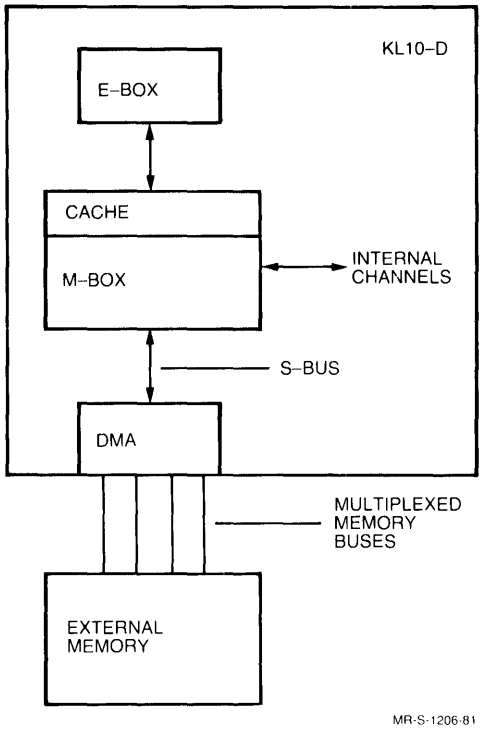


Figure 5–5
External Memory

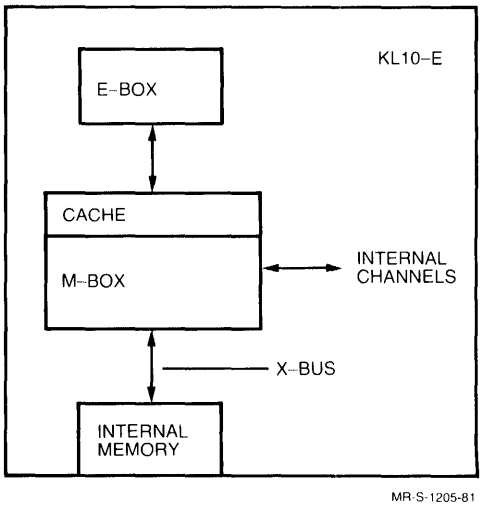


Figure 5–6
Internal Memory

External Memory KL10–D

To meet the requirements of large systems using the TOPS–10 operating system, the KL10–D supports up to 16 external memory modules providing a maximum of 4,096K directly addressable 36-bit words. The DMA supports four memory buses, allowing interleaving and multiple word fetches from separate memory modules. The interleaving is used to increase the bandwidth of TOPS–10-based hardware configurations with external memory. The KL10 reads or writes four words concurrently from memory. These four logically successive memory locations (obtained from four physically separate memory modules) are placed in the Cache Memory. In typical programs, another memory reference need not be initiated until the four instructions or data words have been processed.

Each memory module provides switches that allow a particular module to represent any module of its size in the physically addressable memory space. Thus, one module can replace another without rewiring. The switches also provide for memory interleaving. Table 5–2 lists the features of the external memory system supported by the KL10–D under TOPS–10. The MH10 is currently the only new-build memory of this type offered by DIGITAL.

Table 5–2
External Memory System MH10

Feature	MH10
Word Size	36 bits plus parity
Minimum Memory Size*	64K words
Maximum Memory Size	4096K words
Read Access Time**	
Average	735 ns
Maximum	745 ns
Cycle Time**	1200 ns
Interleaving	1-, 2- or 4-way
Minimum memory unit that can be disabled	64K

* Within CPU Cabinet
**Measured at Memory

Figure 5–7 illustrates the connection and operation of MH10 memory in four-bus/four-way interleaved mode. As an example, assume the E–Box requests a memory reference which is not already in Cache Memory. The KL10–D memory interface (DMA), initiates a four-word memory request to the external memory modules. If properly configured, the request for "word-1" will be recognized by MH10#1; the request for "word-2" will also be recognized by MH10#1, but will be contained in a separate internal module within the MH10 (the MH10 modules are two-way interleaved, internally, and equipped with two sets of read/write electronics so that each MH10 behaves as if it were two "separate" memory modules); the request for "word-3" will be recognized by MH10#2; as will the request for "word-4". Four separate sets of read/write electronics will then begin a read cycle of the four requested words; and, 745 nanoseconds later (maximum), all four words will be placed on the memory buses and sent to the DMA. The DMA will then forward the words to the M–Box where they will be contiguously placed in Cache Memory.

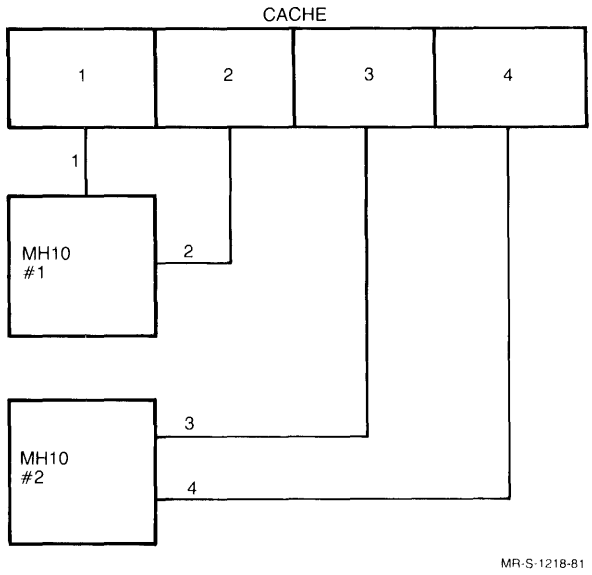


Figure 5–7
MH10 Memory Interleaving Concept

Internal Memory KL10–E

The internal memory implementation of the KL10–E offers improved reliability and lower cost per word than external memories due to its simplified design and, because long bus interfaces are not required, a lower component count. Besides MF20 memory (see MOS memory below), MB20 memory is available for the KL10–E. This memory (Table 5–3) can be configured to contain up to 128K words per package with a single set of read/write electronics. Multiple 128K-word modules can be included in the system package. Each module operates independently of any other. This permits memory references to overlap. MB20 is a single-port memory module. No external direct-to-memory devices are supported on the KL10–E. (All memory requests are directed through and arbitrated by the M–Box in the CPU.) The various memory controllers are interfaced directly to the M–Box through the S–Box. Figure 5–8 illustrates internal memory implementation.

Table 5–3
Internal Memory Systems

Feature	MF20(MOS)
Word Size	36 bits plus 7 error-correcting bits
Minimum Memory Size*	256K words
Maximum Memory Size*	2048K words
Read Access Time**	467 ns
Read Cycle Time**	
1-word	667 ns
4-word	1267 ns
Interleaving	4-way
Minimum memory unit that can be disabled	64K

* Within CPU Cabinet
**Measured at Memory

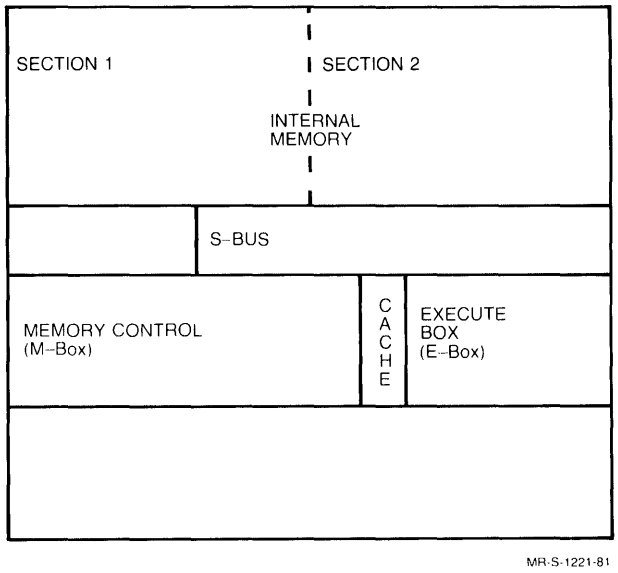


Figure 5–8
Internal Memory Systems Configuration

MOS Memory

The KL10–E is offered with MOS (Metal-Oxide Semiconductor) memory. This memory has a word length of 44 bits (36 data bits, seven error detection and correction bits, and one spare bit). This implementation allows hardware detection and correction of a single bit data error and hardware detection of a double-bit error. MOS memory is available in 265K word increments to a maximum of 3 megawords.

If a correctable (single-bit) error is detected, the data word will be automatically corrected by the hardware and the CPU notified (with a flag) of the event. This design allows the program to continue without consideration of such memory deterioration while, at the same time, it allows the operating system to gather statistics concerning possible failing areas of memory.

If a detectable but noncorrectable (double-bit) error is encountered, the memory controller signals the CPU, sets a noncorrectable error flag, and provides certain other data on the first error detected. The operating system will mark the page containing this location and will not reuse it.

The spare bit can be substituted (upon software command) for failing bits, significantly enhancing the mean time between failure (MTBF) for this type of memory.

Cache Memory

The purpose of cache memory is to decrease instruction execution time by substantially speeding the average memory reference. This is accomplished by placing a high-speed semiconductor memory (the "cache") inside the M–Box. The cache can hold up to 2K words from memory. Whenever the program accesses a word held in cache, the request is satisfied in 133 nanoseconds, compared to 867 nanoseconds or more for a memory reference (measured at the E–Box).

The success of this scheme depends on the quality of the algorithm used to decide which 2K from memory is cached. Cached locations change constantly with varying system demands, but the algorithm is based on the assumption that memory references tend to be somewhat localized. In a typical program the flow of control is linear within a narrow scope. If an instruction has just been executed from location N, there is a high probability that there will soon be an instruction executed from location N + 1.

The "hit rate" of the KL10 cache memory usually exceeds 90 percent.

There are several unique design features of the KL10 cache memory:

- KL10 cache is not a write-through cache. If the E–Box instructs the M–Box to write a given location, the location is modified only in cache. The corresponding physical location will be updated only when the monitor instructs the M–Box to sweep cache, or when a quadword (four adjacent 36-bit words) must be emptied to make room for new data.
- The cache is organized to handle physical addresses. The cache scheme used on some other large systems, however, is oriented toward virtual addresses. The algorithm used in the KL10 was developed by Stanford University, and its modeling demonstrates that the use of physical addresses in the cache algorithm is more efficient than the use of virtual addresses.
- The hardware's use of the cache is dependent upon the M–Box microcode, which is normally set up to support all four cache pages. However, if desired,

some or all of the cache can be turned off. This option is exercised when the front end is initializing the KL10 at system start-up.

Organization

The cache can hold up to 2048 words from memory. It is organized as four separate pages which operate in parallel, each containing 512 words.

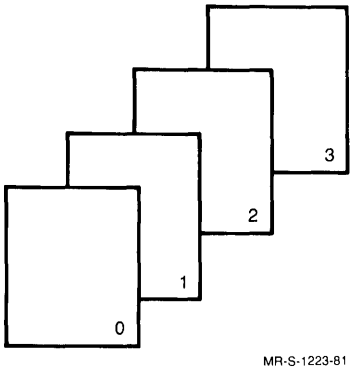


Figure 5–9
Cache Organization

For simplicity, let us consider one of these four pages and the format of the data stored within it. The structure is identical for each of the pages.

Each cache page has a directory associated with it. The directory consists of 128 entries, each entry being 13 bits wide. A single directory entry contains information concerning four words of data within the cache page.

This results in a structure similar to that of Figure 5–10.

A single entry in the cache is represented by Figure 5–11.

A four-word cell described by a single directory entry is called a "quadword." The 13-bit directory entry for a quadword contains the physical page number of the page in memory that originated the quadword. In turn, the position of a word within a cache page is always the same as the position of the word in its original page of memory.

Consider a simple example. Assume that we have only one page of cache and its associated directory, rather than the four that are actually provided in the hardware. Suppose that the E–Box has requested the contents of physical address 14707002 (addresses are in octal notation), a 22-bit address. The M–Box must determine if it must read memory location 14707002, or, if that location is already in cache. The first step is to split the physical address into a 13-bit physical page number, in this case 14707 octal, and a 9-bit index into the page 002 octal In other words, we are concerned with the 002nd word of physical page 14707. If this word is already cached, then the only place it could be in our single cache page would be in word 002 (because "the position of a word within a cache page is always the same as the position of the word in its original page"). The M–Box must examine the 13-bit directory entry corresponding to the 002nd word of the page and compare it to the desired physical page number of 14707. If the directory entry holds 14707, then the 002nd cache page word is indeed the word we are looking for. If the comparison fails, then we have no choice but to read physical memory.

Since there is exactly one directory entry for each quadword, it follows that all four words in the quadword must come from the same physical page. Moreover, a word must have the same position in the cache page that it had in the physical memory page. These facts imply that the four words in a single quadword are physically contiguous in memory as well as in cache. This concept is key to the KL10 cache design.

The previous example was simplified by the omission of three-fourths of the cache pages. In a real system with four pages (2K) of cache, a given physical word

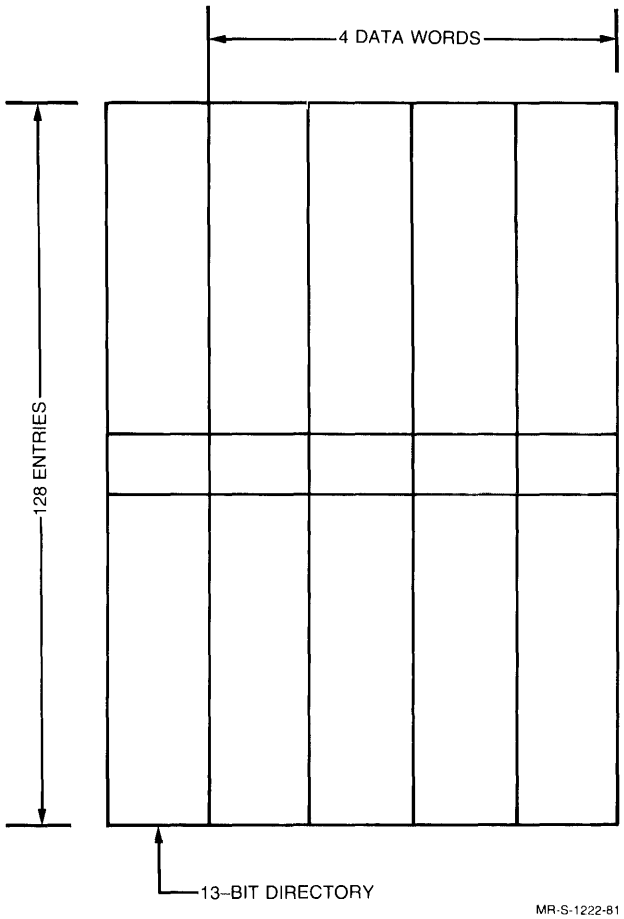


Figure 5–10
Cache Page Structure

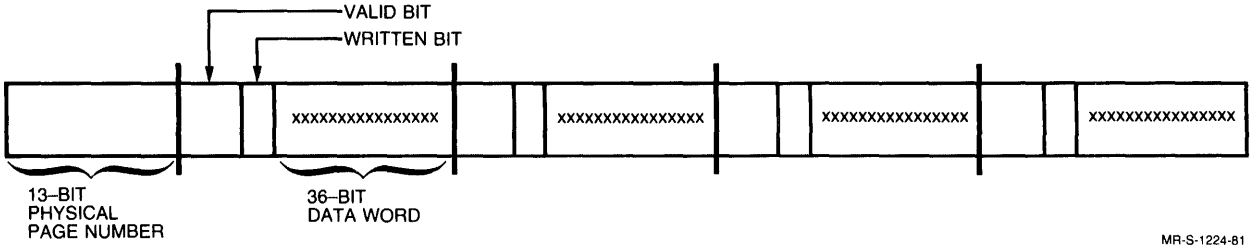


Figure 5–11
Cache Entry

might actually reside in any one of the four pages of cache. For instance, if we need physical address 14707002, we have to keep in mind the physical page number (14707) and the index into that page (002). If the word resides in any of the four cache pages, we know it has to be in word 002 of whichever page holds it, just as we earlier knew that it had to be in word 002 of the single cache page. The M–Box has to compare the desired physical page number of 14707 to the contents of four directory entries, one for word 002 of each of the four cache pages. If a match is found for any one, then the data is taken from the proper page. Otherwise physical memory must be read.

The algorithm that determines the order in which the cache pages are filled uses tables in RAM storage. Therefore, by rewriting the tables it is possible, under program control, to "shut down" part or all of the cache.

System Control of Cache

There are three different operations to which the monitor can subject the cache: invalidation, validation, and unloading. These operations can be performed on the entire cache or on entries belonging to a single memory page.

To invalidate a location means to clear its valid and written bits, which empties the location. Validation of a location means that, if an entry has been written since it was brought in from memory, then the modified contents must be written back into physical memory. This happens because the cache is not a write-through cache. The unloading of a location requires the M–Box to validate the location and then to invalidate it.

Memory Mapping on the KL10

The KL10 provides a hardware implementation of memory address mapping from a program's memory address space (effective address or virtual address) to the physical memory address space. The mapping substitutes the most significant bits of the memory address. Mapping provides access to the entire physical memory space, which can be up to 16 times larger than the maximum user address space. The user's effective address space is 256K words addressed with 18-bit addresses. The physical address space is 4,096K words addressed with 22-bit addresses (4,096K is equivalent to 4,194,304 decimal). Provisions exist in the KL10 to allow user address space larger than 256K words.

The memory-mapping process uses the nine most significant bits of the virtual address as an index into the appropriate user or executive page map. The data located by the index provides 13 bits that are appended to the nine least significant bits of the virtual address in order to form the 22-bit physical address. Also provided are three bits that indicate what type of memory requests are allowed to the page in question (for example, none, read-only, proprietary). Figure 5–12 illustrates this mapping via the page table.

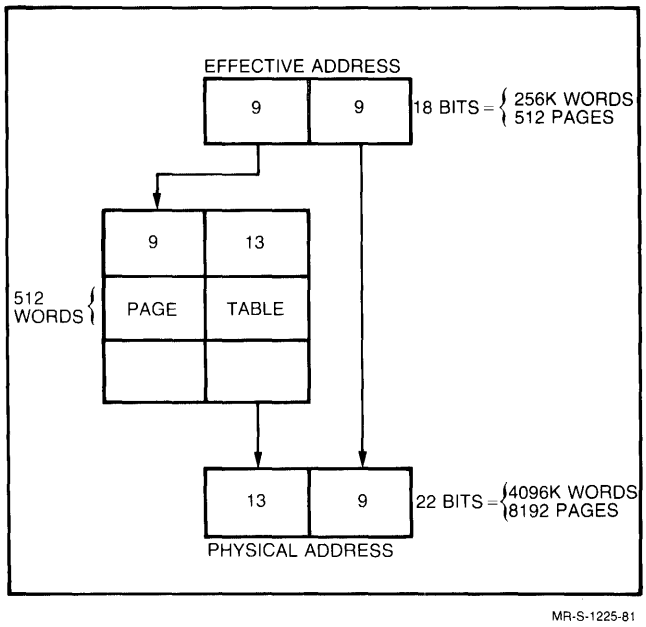


Figure 5–12
Memory Mapping

If the address is in the range 0–17 (octal), inclusive, the hardware fast register blocks are referenced, instead of the main memory system. The user mode bit and the nine high-order bits of the virtual address are used to do a "table look-up" in the hardware page table. If a page table entry exists, the contents of the

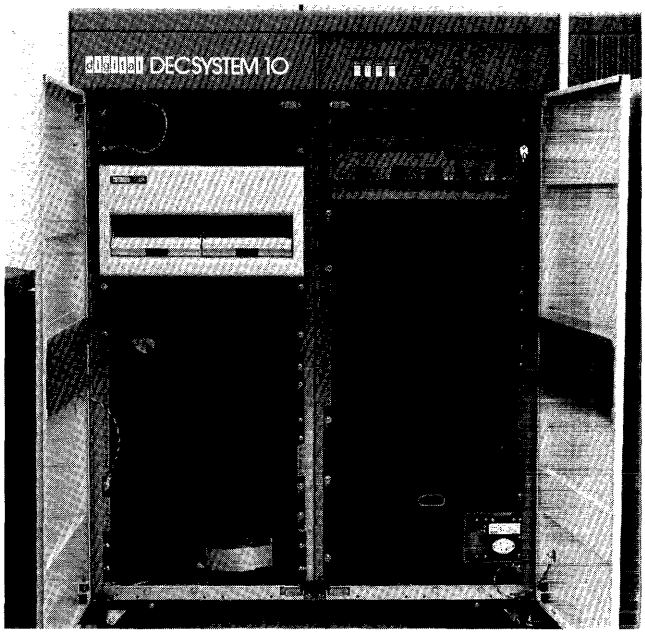
entry supply the 13-bit most significant part of the physical memory address and also supply three bits which indicate types of memory references allowed to this page. If the memory request is consistent with the request type allowed, the physical address used consists of the 13 bits from the related page table entry as the most significant bits of the physical address and the nine least significant bits of the effective address as the least significant bits of the physical address.

When the relocation data for a referenced page does not exist in the hardware page table, the microcode reads the relocation data from the page table in memory, stores it in the hardware page table of the KL10, and proceeds.

The operating system assigns the memory area for each user by loading the various in-memory page tables, setting up the trap locations in the user page map, and responding appropriately when a trap occurs. The monitor provides memory protection for itself and for each user by filling only the page tables with entries that are allowed to be accessed. This makes it impossible for a user to infringe on another user's physical memory area.

The major benefits of this paging capability are:

- Full memory protection of one user from another through hardware and microcode
- Freedom to scatter the pages of a user virtual memory area throughout physical memory (programs do not have to be physically contiguous), thus eliminating memory shuffling and complex memory management even though a user program grows during execution
- The opportunity to execute a program when all of its pages are not in physical memory (that is, a virtual memory capability)



Front-End Subsystem

A key element in the operation and maintenance of the KL10 is the PDP–11-based console/diagnostic front-end computer. This minicomputer provides all console functions for the KL10 and the TOPS–10 operating system.

The PDP–11 interfaces to the KL10 through a dedicated Digital Ten-to-Eleven (DTE) interface with a full capability UNIBUS. The DTE, under control of both the KL10 and the front-end PDP–11, can examine or deposit words into memory and can provide two-way data transfers. Asynchronous communications lines for the console terminal and a dedicated line for remote/local diagnostic functions (KLINIK) connect the user to the operating system by means of the console front end.

In addition to the DTE link, both the KL10 (through an RH20) and the PDP–11 (through an RH11) are interfaced to separate ports of an RP06 Disk Drive. Figure 5–13 illustrates these interconnections.

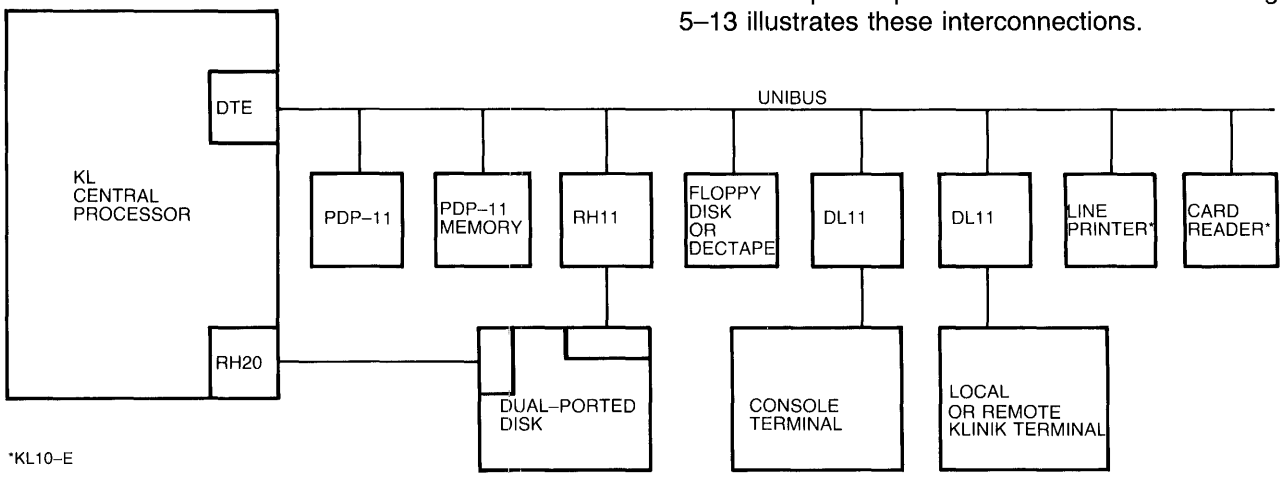


Figure 5–13
Front-end Subsystem

In addition to the console and diagnostic functions, the KL10–E allows unit record and asynchronous communications equipment support by means of the front-end console/diagnostic PDP–11. These devices operate identically (from the user standpoint) to those interfaced to the DECsystem-10s I/O bus. They are explained in detail in the section below on I/O devices.

All KL10-based systems rely on the front end for at least two basic functions. First, the PDP–11 allows the initiation of CPU operations from a dead stop. This process involves setting up KL10 status, loading microcode, configuring memory, and starting the monitor bootstrap. These operations are conducted primarily across a diagnostic bus that connects the front end to both the E–Box and the M–Box by means of the software. The second important function of the PDP–11 is to support the console terminal with which an operator can control the system. It is this terminal that governs the operating system and the tasks running under it. The terminal can also be used as a normal user terminal. Implementation of these functions is accomplished by means of a special operating system that runs in the PDP–11 with supporting code for TOPS–10.

The PDP–11-based operating system varies somewhat between various system models, but is built around Digital's RSX (real-time system) software. This software allows concurrent operation of multiple tasks. One task is the "command parser," the program that recognizes commands typed on the console terminal and passes appropriate messages (through the DTE) to the KL10-resident operating system (TOPS–10). Other tasks include KLINIT, which oversees initialization of the KL10 processor, and KLINIK, which provides a diagnostic environment that permits diagnosis and control of the KL10 to privileged users from either local or remote locations.

Devices associated with this front end which support console and diagnostic operations, but which can not be used as peripheral devices, include the RH11 disk controller and either a floppy disk drive or DECtape drive. Either can be used as an alternate bootstrap device if, for some reason, the RP06 disk cannot be used or contains incorrect information.

Front-end operations require that the dual-ported RP06 be available for normal operations. The RP06 is connected to both the KL10 and the PDP–11 through dual-ported hardware. The disk used is KL10-formatted. There are both hardware and software interlocks to prevent the KL10 and PDP–11 from interfering with one another. The PDP–11 disk area is not available for user storage.

KL10 systems can support up to four PDP–11s; however, only one of these can be the controlling front end. In order to prevent conflicts between different PDP–11s, the operations described in this section can only be accomplished through a "privileged" hardware- and software-selectable DTE.

More about DTE operations is included in the next section

Input/Output Subsystem

The KL10 input/output subsystem provides standard interfaces and control logic to allow the KL10 to be connected to a wide range of peripheral and communications equipment. These peripherals include disks, magnetic tapes, card readers, line printers, plotters and, through PDP–11-based communications subsystems, synchronous and asynchronous lines. Details of the operation and the specifications of these peripheral devices are available from your Digital sales representative.

To standardize the interface and permit the connection of a wide range of peripheral devices, Digital has long used a "bus" architecture. Three external buses (see Figure 5–14) are available to communicate with the KL10 CPU:

- Multiplexed I/O bus
- MASSBUS
- UNIBUS

Each is discussed in detail in the following paragraphs.

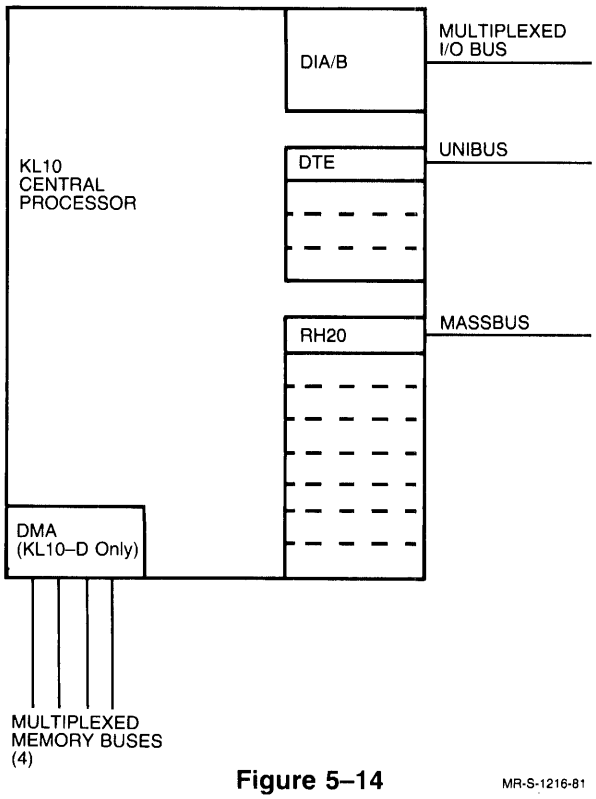


Figure 5–14 MR-S-1216-81
KL10 Input/Output Bus Architecture

Multiplexed I/O Bus

A multiplexed I/O bus is provided on all KL10–Ds and is optional on KL10–Es. The KL10 I/O bus is supported by the TOPS–10 operating system. The I/O bus provides control and a character-by-character, interrupt-driven interface for low-speed devices. Under TOPS–10, line printers, card readers, etc., are supported through the I/O bus. Figure 5–15 illustrates the hardware associated with the KL10 I/O bus.

Multiplexed Memory Bus Subsystem

An external memory bus system is supported on the KL10–D CPU, allowing external memory to be connected to the processor. The KL10–E does not provide external memory because its memory is internal. The DMA–10 (shown in Figure 5–16) allows up to sixteen external memory modules and up to 4,096K words of external memory. The memory bus system operates completely asynchronously allowing support of differing memory speeds. Four buses are provided allowing four-word concurrent fetch on four-way interleaved memory systems. These external memory modules may also be interfaced to external direct-to-memory channels and controllers, for tapes and communication subsystems.

MASSBUS

DIGITAL's MASSBUS allows connection of a wide range of DIGITAL-supplied disks and magnetic tapes to the KL10 central processor. (See Section 7 for information about MASSBUS peripheral devices.) Each MASSBUS can interface as many as eight peripheral device controllers to a single RH20 MASSBUS controller. Each RH20 interfaces to the E–Box via the E Bus and to one of eight data channels located in the M–Box via the channel bus. (See Figure 5–17.)

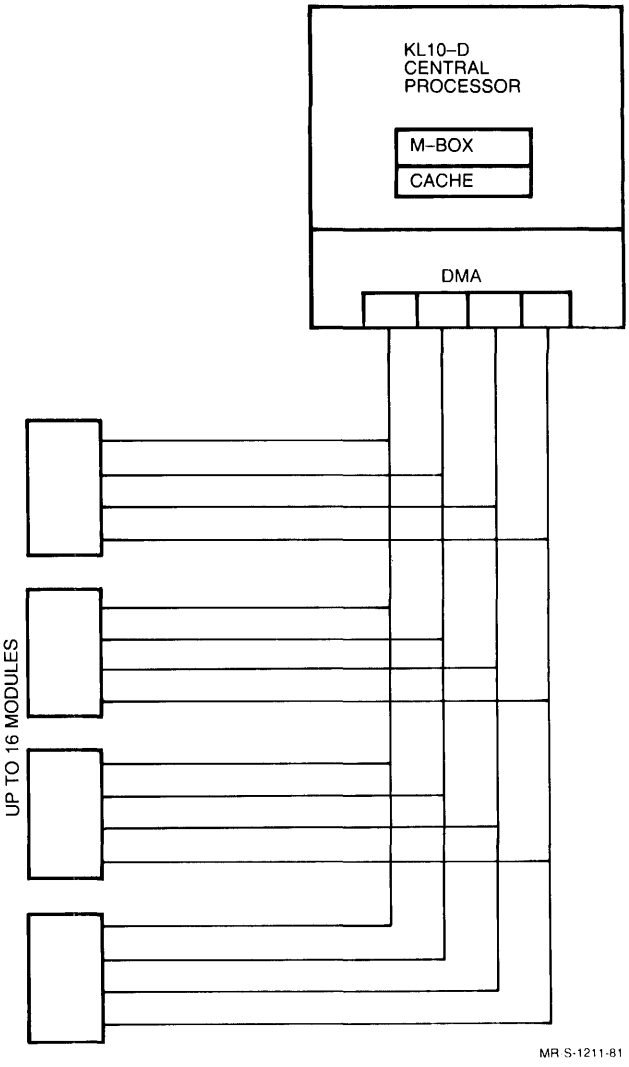


Figure 5–16
KL10–D Memory Bus Structure

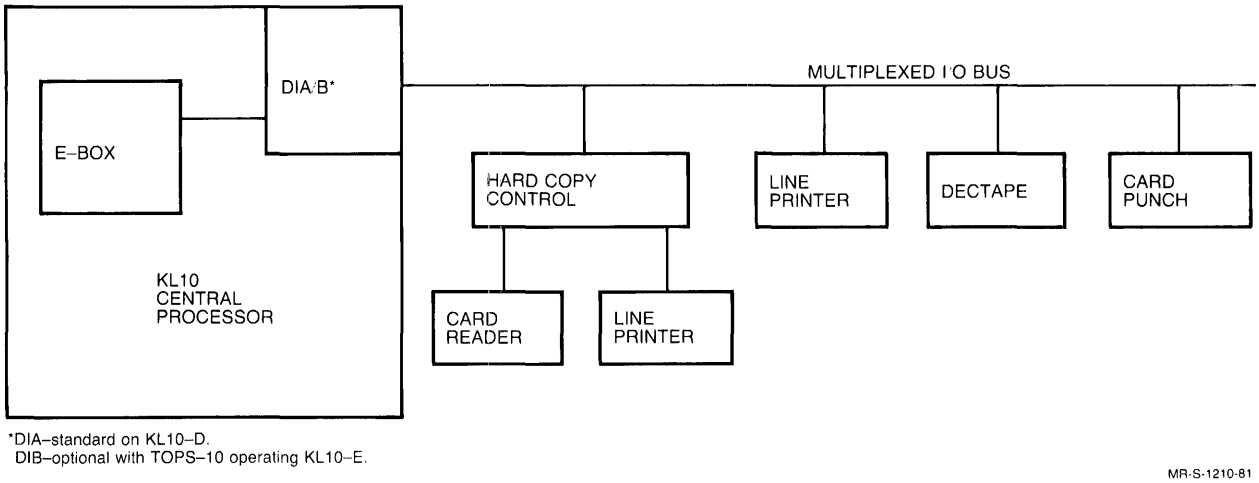


Figure 5–15
Multiplexed I/O Bus Architecture

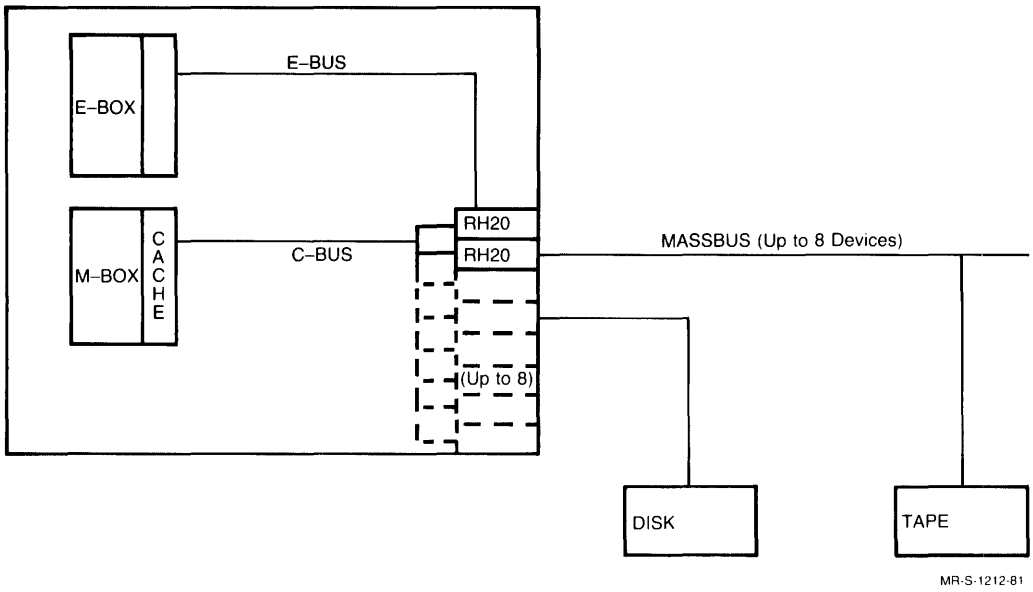


Figure 5–17
MASSBUS Interface

The channel also includes a series of multiple-word channel buffers to support high-speed I/O between the KL10 memory controller and individual MASSBUS devices. Up to eight RH20s can be installed in the KL10. For each device, the channel provides a 15-word data buffer, a channel command word register, and a control command word pointer that serves as a program counter. The channels transfer data (independently of the KL10 execution logic) by executing channel programs placed in memory by device service routines (in the operating system) that use memory cycles.

The RH20 integrated controllers are interfaced to the memory control unit (M–Box) of the KL10 CPU via a channel bus. The bus is a physically short high-speed data transfer path designed for peak I/O bandwidth of six million words per second (well in excess of that encountered in any existing memory or peripheral units) between the memory control unit and the MASSBUS controllers (RH20s). Operating in synchronous time-division multiplexed mode, the channel bus allows the connection of multiple RH20s in memory.

Each RH20 terminates in a MASSBUS. This high-speed data path connects the RH20 integrated controllers and mass storage devices such as disk or tape. Operating either asynchronously or synchronously, the MASSBUS transfers control and status information or blocks of data between devices and controllers.

DIGITAL-supplied disks and tapes connect directly to the MASSBUS, providing an integrated high-speed subsystem for mass storage.

UNIBUS

DIGITAL's UNIBUS provides the third path between external devices and the KL10 CPU. As illustrated in Figure 5–18, this interface utilizes the DTE interface. The DTE provides:

- The ability to examine or deposit words in PDP–11 memory and specified areas of KL10 memory (under the control of a PDP–11 processor connected to the DTE-terminated UNIBUS or the KL10 CPU).
- High-speed simultaneous two-way transfer of data between PDP–11 and KL10 memory. This transfer operates in a PDP–11 "byte" mode and transfers eight- or 16-bit bytes under the control of either the PDP–11 or KL10 processors.
- Doorbell interrupts. The PDP–11 can cause an interrupt to the KL10 processor and vice versa.

The DTE's two operating modes, restricted and privileged, are controlled by a hardware switch on the DTE. The privileged DTE is used only for the console/front-end PDP–11. Only the restricted DTE, which is used to control peripheral devices and communications equipment, is described here.

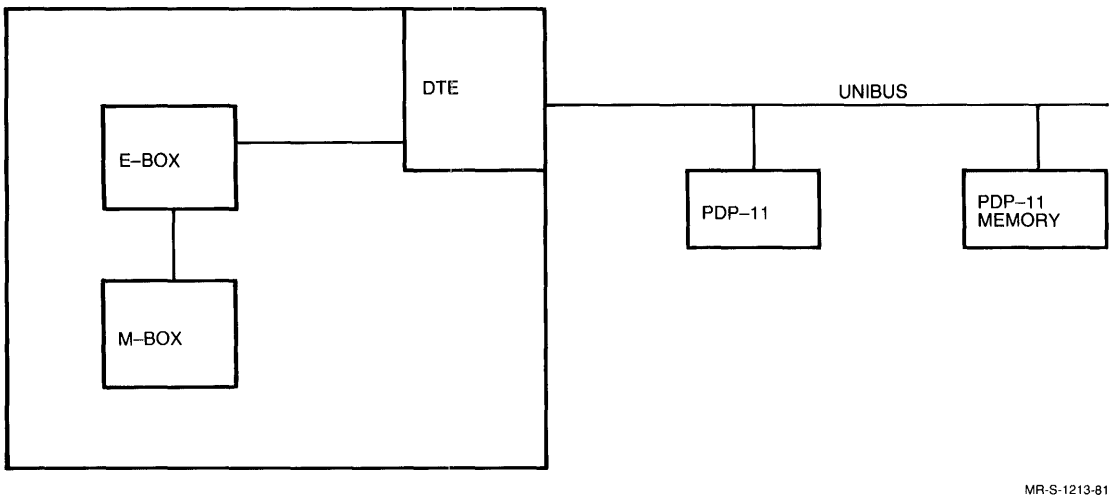


Figure 5–18
UNIBUS Interface

There are two ways in which the DTE allows a PDP–11 to communicate with KL10 memory. First, the PDP–11 can use an examine/deposit feature that permits the PDP–11 to read or write a single KL10 word. The other method is with byte transfers in which the DTE is responsible for transferring a string of data to or from KL10 memory. It is important to note that the PDP–11 memory is itself a UNIBUS device, and the DTE can directly access it. For KL10 accesses, the DTE interfaces to the E–Box of the KL10 processor. Of course, the DTE is totally inoperative unless a PDP–11 is connected to the DTE and the appropriate program is running in the PDP–11.

Examines and deposits by the PDP–11 can always be made to any address within windows defined in KL10 memory. The windows are specified by the operating system executive process table. There are two windows, the "to-10" area and the "to-11" area. These differ in their availability to the two processors as follows:

	Can -10 write?	Can -11 write?
to -10 area	Yes	Yes
to -11 area	Yes	No

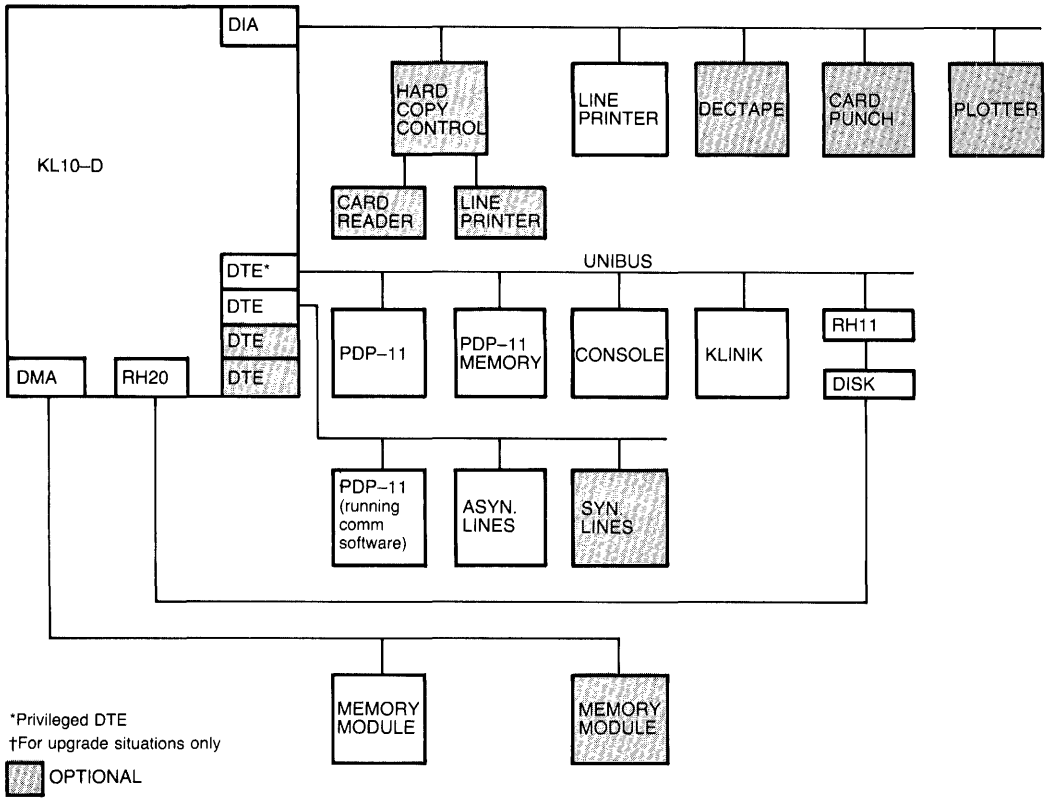
A restricted DTE cannot examine or deposit outside of the windows established by the operating system. This enables the KL10 to protect itself from a malfunctioning PDP–11. However, a privileged front-end (one used only for the console/front-end processor) can examine and deposit anywhere in KL10 memory, allowing for diagnostic functions.

The other transfer mechanism, byte transfer, permits transfers to or from anywhere in KL10 memory. Byte size can be 8 or 16 bits at the program's option, and byte transfer supports concurrent "to-11" and "to-10" transfers.

Once a transfer has been initiated, the DTE handles it without further intervention from either CPU at the program level; the operating system will not be interrupted until the entire transfer is complete. The DTE recognizes the end of the transfer by the expiration of a byte counter. Transfer completion results in an interrupt on the assigned priority interrupt level. On the KL10 side, actual implementation of a byte or word transfer is by microcode. Thus, during transfers through the DTE, cycles stolen from the microcode incur some of CPU overhead.

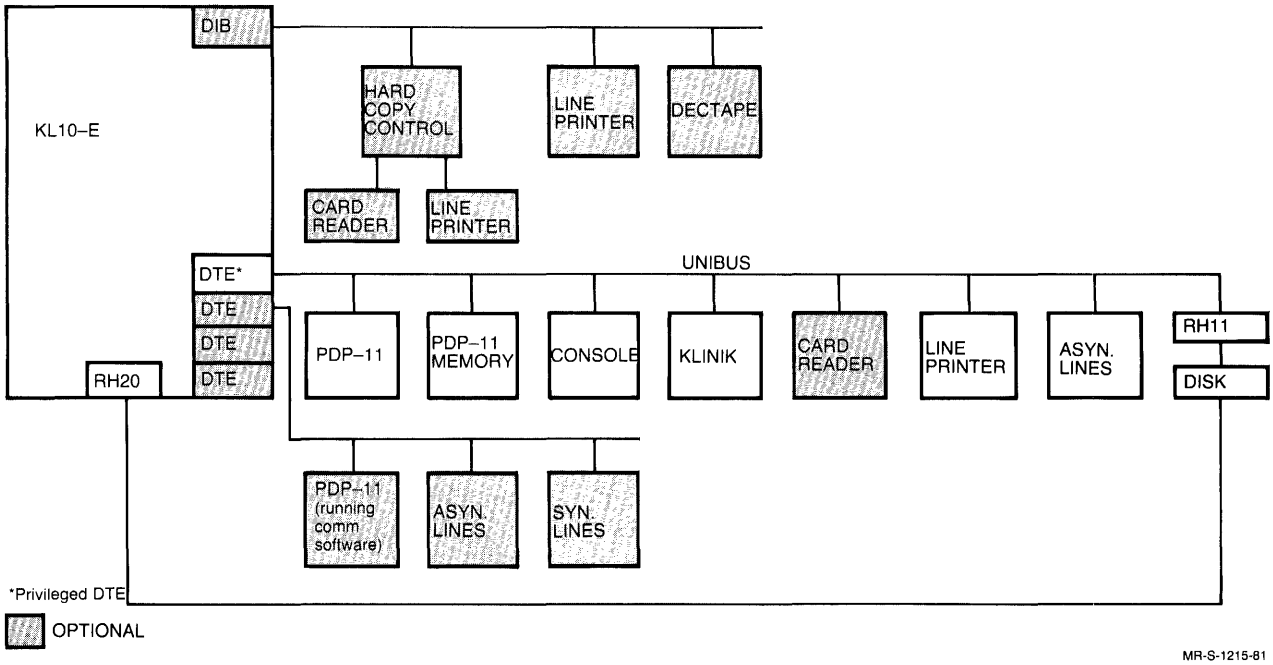
As indicated earlier, DTE operation requires a PDP–11 processor. While service routines are integral to the operating system, it is the PDP–11 program that actually handles peripheral equipment and the control of its operation. Card readers and lineprinters are supported on the console/front-end processor, as are a limited number of asynchronous communications lines. Figures 5–19 and 5–20 illustrate UNIBUS and I/O bus peripheral devices.

Additional information on any peripheral or communications subsystem is available from your DIGITAL representative.



MR-S-1214-81

Figure 5–19
KL10–D Unibus and I/O Bus Peripheral Devices



MR-S-1215-81

Figure 5–20
KL10–E Unibus and I/O Bus Peripheral Devices

6
The KS10 Central Processor

The KS10 central processing unit (CPU) forms the basis of the DECSYSTEM–2020.

The KS10 has changed mainframe computing. People constrained by budgets to settle for something less than a mainframe can now install a KS10-based DECSYSTEM–2020 with full assurance that software-compatible growth is easy, cost-effective, and straight-forward.

The DECSYSTEM–2020 comprises a number of major units or subsystems that communicate with each other. The minimum system has five subsystems: processor, MOS memory, console, a UNIBUS adapter to handle the disk system, and a UNIBUS adapter to handle all other peripheral devices.

The console, which is based on a microprocessor, boots the system from disk and handles interaction of the operator or a remote diagnostic link with other subsystems.

SYSTEM ARCHITECTURE

The KS10 central processor forms the basis for the DECSYSTEM–2020 computer system. The DECSYSTEM–2020 consists of four major subsystems:

- KS10 CPU and memory
- Console and diagnostic microprocessor
- UNIBUS adapters
- Peripheral controllers

KS10 Technology

To meet its goal of full TOPS–10 functionality at low cost, the KS10 was designed as a microprogrammed central processor using low-power Schottky TTL and featuring 4-bit data path slice microprocessor.

Microprogramming technology used in the KS10 design relied heavily on experience gained in the design and implementation of the successful KL10 central processor and several of DIGITAL's mini/midi central processors. Microcoded machine instructions are implemented by one or more microstore instructions; that is, to accomplish a given machine instruction, the microinstruction sequence associated with the instruction moves or operates on data or controls steps necessary to achieve the desired effect of the instruction.

Microprogramming implementation of the KS10 central processor provides several important advantages:

- Engineering changes or central processor improvements can often be microstore (firmware) changes rather than wiring changes.
- The packaging technology of microcoded hardware reduces machine size and increases flexibility of machine instructions.
- Virtual-address cache and virtual memory control operations are speeded by microprogramming.

In addition to the microprogramming implementation of the KS10, the 4-bit-slice technology allows the KS10 to be reliable, low in cost, flexible, and compact. Chip technology has progressed to the point where an entire data path, including data registers, data operations, and testing and control functions for 4 bits, is available on a single chip. Thus, to implement a 36-bit KS10 central processor, nine 4-bit data path slice chips are used (plus a tenth 4-bit data path slice chip for "extra" control bits). Control lines from each are connected to the microstore logic so the microcode can dictate function, control, and data flow throughout the system.

The KS10 Central Processing Unit

The KS10 central processing unit consists of four extended hex modules:

- Data Path Executive (DPE) Module — contains data path, registers, cache, PI system, and Dispatch ROM (Read-Only-Memory).
- Data Path Memory (DPM) Module — contains bus interface, processor status flags, paging, cache directory, and shift counter.
- Control RAM (Random Access Memory) Address (CRA) Module — contains next microcode address logic, microcode leading hardware, and 2K x 36 bits of microcode.
- Control RAM Memory (CRM) Module — contains 2K x 60 bits of microcode.

The heart of the KS10 is an internal backplane bus. Called the KS10 bus, it provides a control and data path between the processor, memory, console, and peripheral devices.

Figure 6–1 illustrates the interconnection of the KS10 central processor, memory, UNIBUS Adapters, and console via the KS10 bus.

This bus is a multiplexed two-cycle bus over which command and address information is transmitted from one bus device to another during a bus cycle. Data is then transferred between the addressed device during the following bus cycle. The bus operates in a bus request/bus grant mode with a bus cycle of 150 nanoseconds.

The KS10 bus is a synchronous backplane bus internal to the KS10 processor. The bus can accommodate up to eight devices (including CPU, memory, and console) and performs several major functions:

- Transfers data to and from MOS memory via the memory controller under control of the CPU, console, or a UNIBUS Adapter.
- Transfers data to and from I/O device registers under control of the CPU or console. (Any device external to the CPU is considered an I/O device.)
- Transmits priority interrupt requests generated by the UNIBUS Adapters, and upon CPU request, provides device numbers and interrupt vectors from the interrupting device to the CPU.
- Provides a continuous clock train that is used by all devices to sequence logic and to synchronize operation with the rest of the system.
- Allows the console to reset the system, perform diagnostic functions, and to signal AC power failure to the devices on the bus.

The KS10 bus data path is 36 bits wide. Additionally there are two parity bits associated with the data lines: one for data lines 0–17, and a second for data lines 18–35. Each device checks for correct (even) parity

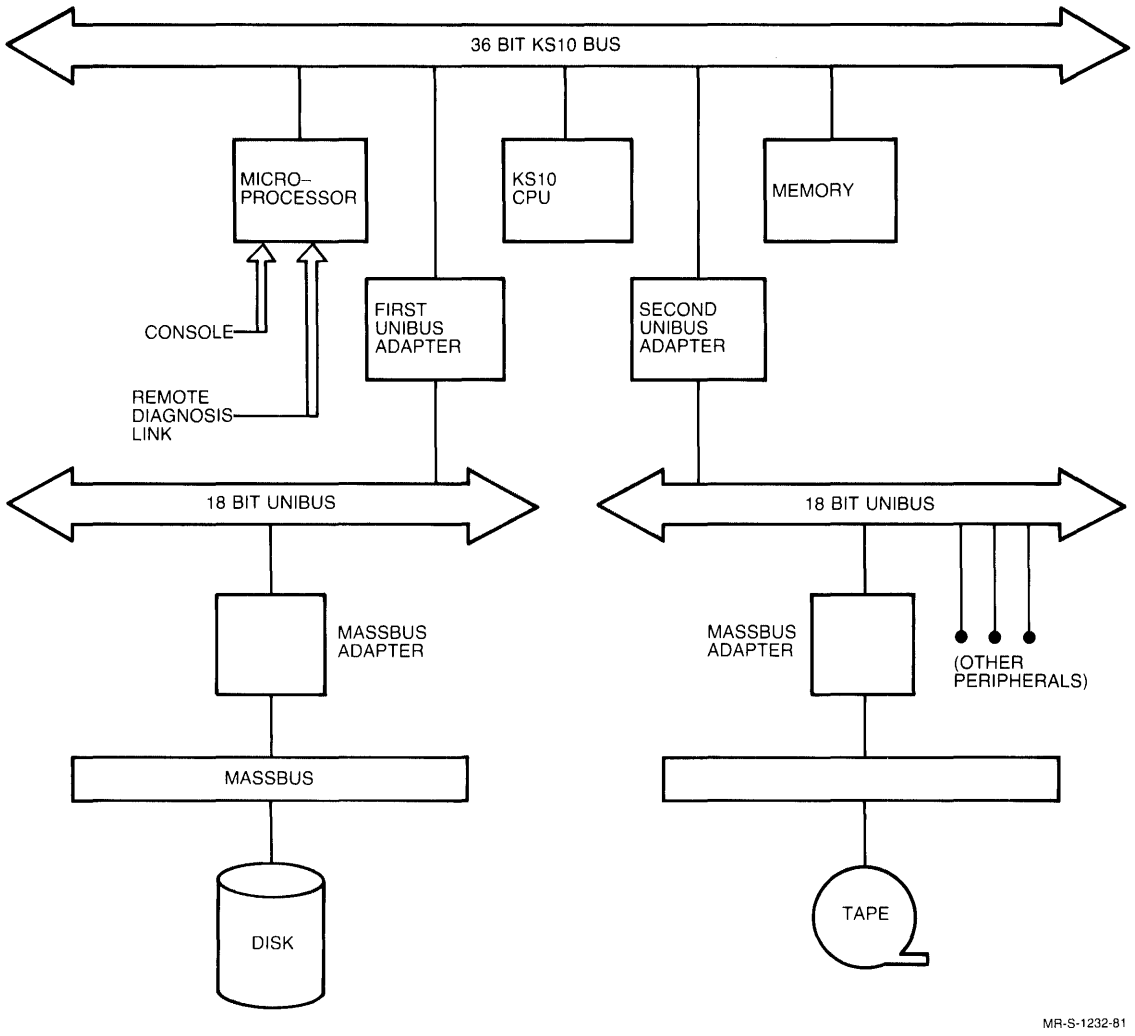


Figure 6–1
KS10 System Architecture

when it receives information over the bus. If bad parity is detected, the CPU clock is stopped.

Command/address information is transmitted over the bus during a command/address cycle. Then, during the following bus cycle, the same bus wires are used to transmit the 36-bit data word during the data cycle.

Before any device can transfer information over the KS10 bus, it must first request and be granted the bus. There is a bus request line and a corresponding grant line for each device. The bus arbitrator on the console module monitors all requests, resolves priority, and, when the bus is free, grants the bus by asserting the grant line for the highest-priority requesting device. The bus arbitrator can handle up to eight devices.

Cache Memory

The KS10 cache memory implementation includes 512 words of high-speed memory. Cache memory provides faster effective memory access time than the slower main memory. On typical in-line programs the desired memory reference is found already in cache approximately 80 percent of the time, eliminating a 900-nanosecond main memory access. With a cache access time of 300 nanoseconds and the typical cache hit rate of 80 percent, the effective memory access time for the KS10 is 420 nanoseconds. The cache memory is located on the Data Path Executive (DPE) module in the KS10.

General Registers

The general registers consist of eight accumulator (AC) blocks of 16 words each, located on the Data Path Executive (DPE) Module. The eight sets of registers can be used as accumulators, index registers, or the first sixteen locations in memory. How the registers are used depends upon how they are addressed

in the instruction word. Access time to general registers is 300 nanoseconds.

Microstore

The microstore is located on the Control RAM Address (CRA) and Control RAM Memory (CRM) Modules. 2048 96-bit words are provided. The microcode and microstore are considered integral to the hardware and are not available for operating system or user program use.

Instruction Set

The KS10 central processing unit features 396 microprogrammed instructions. Each instruction is a series of microinstructions that performs various logical functions, such as processor state control, data path control, and actual implementation of each instruction in the KS10's set. Since the instruction set provides so many instructions, fewer instructions are typically required to perform a given function. However, the instructions are logically classed and consistent in mode, making their use straightforward.

All instructions are capable of directly addressing 256K words without resorting to base registers, displacement addressing, or indirect addressing. Instructions can use indirect addressing with indexing to any level. Many instruction classes, including floating point, allow immediate mode addressing where the result of the effective address calculation is used directly as an operand to save storage and speed execution.

In all instructions the nine high-order bits specify the operation and bits 9–12 usually address an accumulator (but are sometimes used for special control purposes such as addressing flags). The rest of the instruction word always supplies information for calculating the effective address. The effective address can be used as an operand (immediate mode) or as the actual (virtual) address to fetch an operand or alter program flow. Bit 13 specifies the type of addressing (direct or indirect); bits 14–17 specify an index register used for address modification (zero indicates no indexing). The remaining eighteen bits (18–35) contain a memory address.

Figure 6–2 illustrates the instruction format of the KS10.

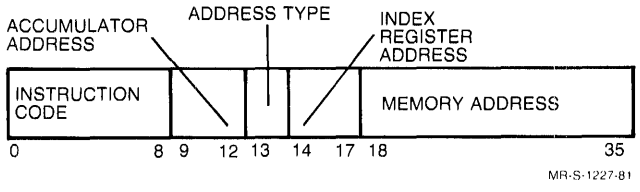


Figure 6–2
KS10 Instruction Format

Half-Word Data Transmission

Half-word data transmission instructions move a half-word and can modify the contents of the other half of the destination location. These 16 instructions differ in the direction they move the chosen half-word and in the way they modify the other half of the destination location.

Full-Word Data Transmission

Full-word data transmission instructions move one or more full words of data from one place to another. These instructions can perform minor arithmetic operations such as calculating the negative or the magnitude of the word being processed.

Byte Manipulation

Five byte manipulation instructions pack or unpack bytes of any length anywhere within a word.

Logic Instructions

Logic instructions shift, rotate, and execute the 16 Boolean operations on two variables.

Fixed-Point Arithmetic

The KS10 is a 2s' complement machine in which zero is represented by a word containing all zeros. There are instructions to scale, negate, add, subtract, multiply, and divide in single- and double-precision floating-point format. In the single-precision floating-point format, one bit is used for the sign, eight bits for the exponent, and 27 bits for the fraction. In double-precision floating-point format, one bit is used for the sign, eight bits for the exponent, and 62 bits for the fraction.

Fixed-Floating Conversion

Special KS10 instructions provide the capability of converting fixed-point formats to or from floating-point formats. Two sets of instructions are provided to perform this function, one set optimized for FORTRAN and a second set optimized for ALGOL.

Arithmetic Testing

The arithmetic testing instructions can cause a program jump or no jump, or a program skip or no skip, depending on the result of an arithmetic test, and may first operate arithmetically on the test word.

Logical Testing, Modification, and Skip

A group of instructions modify and/or test using a mask, and they cause a program skip or no skip, depending upon selected bits in an accumulator.

Program Control

Program control instructions include several types of program jump instructions and subroutine control instructions which jump as well as save or restore information on a stack.

Within the KS10 instruction set there is a group of operation codes that are used as monitor calls. If these instructions are executed by a user program, the program will not execute the instruction, but rather will trap to the operating system. The operating system can then examine the instruction and take appropriate action. This monitor call implementation is basic to the operation of TOPS–10. In a multitask environment, the operating system must control certain operations. This is fully possible through the monitor call (or processor trap) instruction classes.

Business Instuction Set
There are five classes of instructions in the Business Instruction Set of the KS10 central processor. Four of these are arithmetic instructions to add, subtract, multiply, and divide using double-precision fixed-point operands. The STRING instruction is the fifth class, and can perform nine separate string functions.

These functions include an edit capability, decimal-to-binary and binary-to-decimal conversion in both offset and translated mode, move-string in both offset and translated mode, and compare-string in both offset and translated mode. Offset mode is byte modification by addition of the effective address of the string instruction. Besides providing the translation function, these instructions can control AC flags and can detect special characters in the source string.

The Business Instruction Set provides faster processing because there are special instructions for a wide variety of string operations. These instructions can be used on a variety of code types, such as ASCII and EBCDIC.

Trap Handling
The KS10 provides programmed trap instructions to directly handle arithmetic overflows and underflows, pushdown list overflows, and page failures. This trap capability avoids recourse to the program interrupt system.

Fast Register Blocks
Eight sets of 16 fast, general purpose register blocks (accumulators) are provided in the KS10 architecture. They can be used as accumulators, index registers, or as the first 16 locations of memory. Since register addressing is included in the basic instruction format, no special instructions are needed to access these registers. Different register blocks are used for the operating system and individual users, eliminating the need for storing register contents when switching from User Mode to Executive Mode.

Processor Modes
Instructions are executed in one of two modes depending on the state of the mode bit. Programs operate in either User Mode or Executive Mode. In Executive Mode operations, all implemented instructions are legal. The monitor operates in Executive Mode and is able to control all system resources and the state of the processor. In User Mode operations, certain instructions such as direct I/O, are illegal, causing a trap to the monitor. Users are required to issue monitor calls for system services such as I/O.

Memory
The KS10 central processor implementation supports a hierarchical memory system consisting of fast register blocks, high-speed cache memory, main (MOS) memory, and mass memory subsystems such as disks. The fast register blocks, cache memory, and main (MOS) memory are directly controlled by the KS10 hardware and firmware; accessing is microstore controlled. Disk memory access is controlled by the operating system.

Memory Address Mapping
The memory address hardware was developed in conjunction with the software so that memory management is totally transparent to the user. Physical memory is divided into 512-word segments called pages. All addresses, both monitor and user, are translated from the program's virtual address space to the physical memory address space. This facilitates protection of the monitor and allows efficient use of physical memory. For example, only a portion of the monitor need be permanently resident in memory, resulting in more memory available to the user.

The high-order nine bits of an 18-bit virtual address constitute the virtual page number, and are used to index into a hardware page map. If the 13-bit physical page number is found in the hardware page map, then the low-order nine bits of the virtual address are appended to the 13-bit physical page number to form the complete physical memory address. If the entry in the hardware page map does not exist, the memory processor gets the entry from the individual page map in memory, and updates the hardware page map.

There is a page map in memory for the monitor and one for each user. These maps contain storage address pointers which identify a page either in memory or on disk. The system is capable of interpreting three types of address pointers. The first is called a "private"pointer and contains a physical storage address. If this is a memory address, the system loads the hardware page table with the information and completes the requested reference. If it is any other address, the system initiates a trap to the monitor for appropriate action. The second type of address pointer is called a "shared" pointer, and contains an index into a system table at a fixed location. This system table is called the Shared Page Table (SPT) and

contains the physical storage address of the shared page. This table which is addressed through a hardware register, enables the software memory management routines to maintain just one pointer to a page no matter how many processes are sharing the page. The third type of address pointer is called an "indirect" pointer and contains a page number and SPT index. The SPT index is used to index the SPT table to pick up an address of a page table. This page table is indexed by a page number given in the indirect pointer to obtain the physical storage address. This pointer allows one page to be exactly equivalent to another page in a separate address space.

Information pertaining to how long a page has been in memory and the number of processes sharing it is also stored by the hardware into the Core Status Table to aid the monitor in memory management.

TOPS–10 Paging

The TOPS–10 operating system utilizes the page maps to create one- or two-segment programs in roughly the same fashion as was accomplished by protection and relocation registers in earlier processors. However, the KS10 has the advantages of allowing smaller units of memory (512 words instead of the 1024 words in earlier processor designs) the freedom to scatter the pages of a segment randomly in memory to avoid fragmentation and overhead of repacking memory, and the opportunity to execute a program when all of its pages are not in physical memory (i.e., a demand-paged virtual memory capability).

MOS Memory

The main memory subsystem of the KS10 central processor consists of MOS (metal-oxide silicon) memory which is connected to the KS10 bus via a memory controller. The MS10 memory is available in 64K word increments up to a maximum of 512K words. The memory word length is 43 bits: 36 data bits and seven error detection and correction bits. Memory interleaving is not required.

Semiconductor memory offers the advantage of higher speed than core memories. It also offers ease of interfacing to control "outside-world" electronics. The processes used to construct the storage elements within the memory array itself can also be used for addressing and decoding electronics and for output buffering on a single chip. Fewer connections mean higher reliability and lower cost.

In the MOS memory implementation, the storage element takes the form of an MOS "cell" or transistor. In an elementary operation, data bits are stored by charging or not charging the effective capacitance of each storage cell. Reading entails sensing the presence or absence of charge. Only a few hundred electrons of charge differentiate the storage of a "1" and a "0."

Because there is capacitive leakage, this charge must be periodicaly renewed or refreshed. The required frequency of the refresh cycle depends on leakage of the circuit, which increases with temperature. The 128 rows of KS10 MOS memory are automatically refreshed by the memory hardware, one row every 15 microseconds.

All information stored in MOS memory is lost if power is removed from the memory circuitry.

The MS10 is a single-port memory using MOS random-access memory (RAM). The MS10 subsystem provides for single-bit error correction and double-bit error detection. If the memory control logic detects a correctable error on a memory read cycle, the following occurs:

- If the error is in one of the 36 data bits, the error bit and the parity bit are automatically corrected.
- If the error is in one of the seven check bits, no correction is required; an error flag is raised; and the address of the failing memory is held in a register.
- A flag is raised indicating that the logic has corrected an error.
- The memory control logic saves a 20-bit address (for a read or read-pause-write request) and the ECC code for the first error detected to allow software monitoring of memory condition.

MS10 Reliability, Availability, Maintainability, and Performance Features

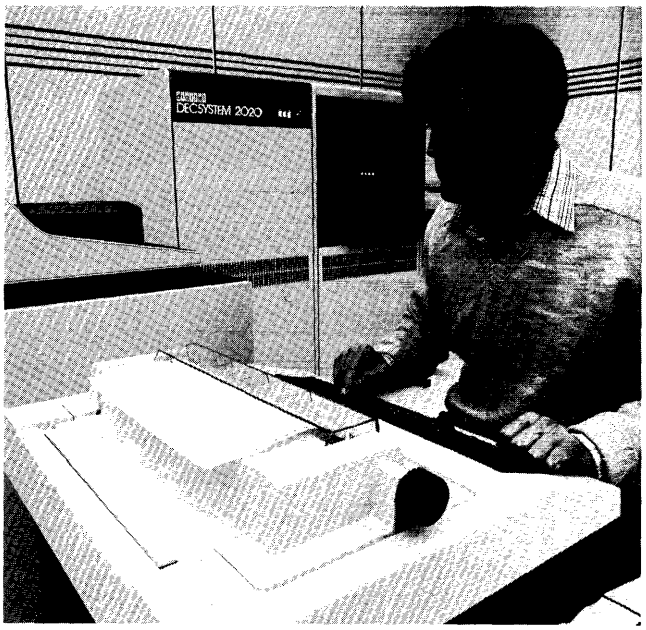
High reliability is a primary goal of the MS10 memory. The following summarizes MS10 reliability, availability, maintainability, and performance (RAMP) features.

- Single-bit error detection and correction and double-bit error detection are implemented in the memory control logic.
- Power supply is a high efficiency switching regulator for low internal power dissipation.
- Storage array component layout facilitates airflow over the module.
- Extensive airflow and temperature profile measurements have been taken.
- Modules are "burned in" prior to installation by manufacturing.
- Isolation of fault to board level.
- Correctable read errors are automatically handled by the hardware. So, correct data is always sent to the CPU, and notification of the (corrected) error is provided.

Console Subsystem

The console is an extremely important subsystem in the implementation of the KS10 central processor because it controls all console and diagnostic functions. The console is housed on a single extended hex module and uses an eight-bit microprocessor. To program the microprocessor, an 8K programmable read-only memory (PROM) and 1K random- access memory (RAM) are provided. Two universal synchronous/asynchronous receiver/transmitter (USART) interfaces are provided, one for console operation and one for remote diagnosis link operation. The remote diagnosis link and the microprocessor code make possible remote diagnosis of the system.

The console is interfaced to the KS10 bus for data and control functions. In addition, direct connections to other KS10 CPU modules are provided to allow the console to perform test and housekeeping functions.



The local operator controls the KS10 with a set of commands typed at the console terminal (CTY). The CTY connects directly to the microprocessor-based console hardware over a serial line interface (USART). A second serial line operating in parallel with the first can also be connected to the console hardware, allowing the KS10 to be controlled by a remote diagnosis link.

Commands typed at the CTY or entered from the remote diagnosis link are implemented by a program running in the console's microprocessor. This program, resident in PROM, automatically runs when power is turned ON. The console program resident in PROM is not destroyed by powering down the system.

Figure 6–3 illustrates the console subsystem.

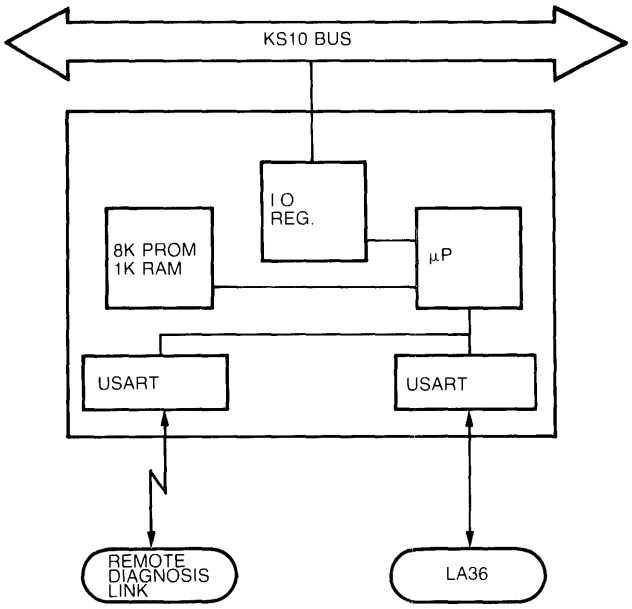


Figure 6–3
KS10 Console Subsystem

MR-S-1228-81

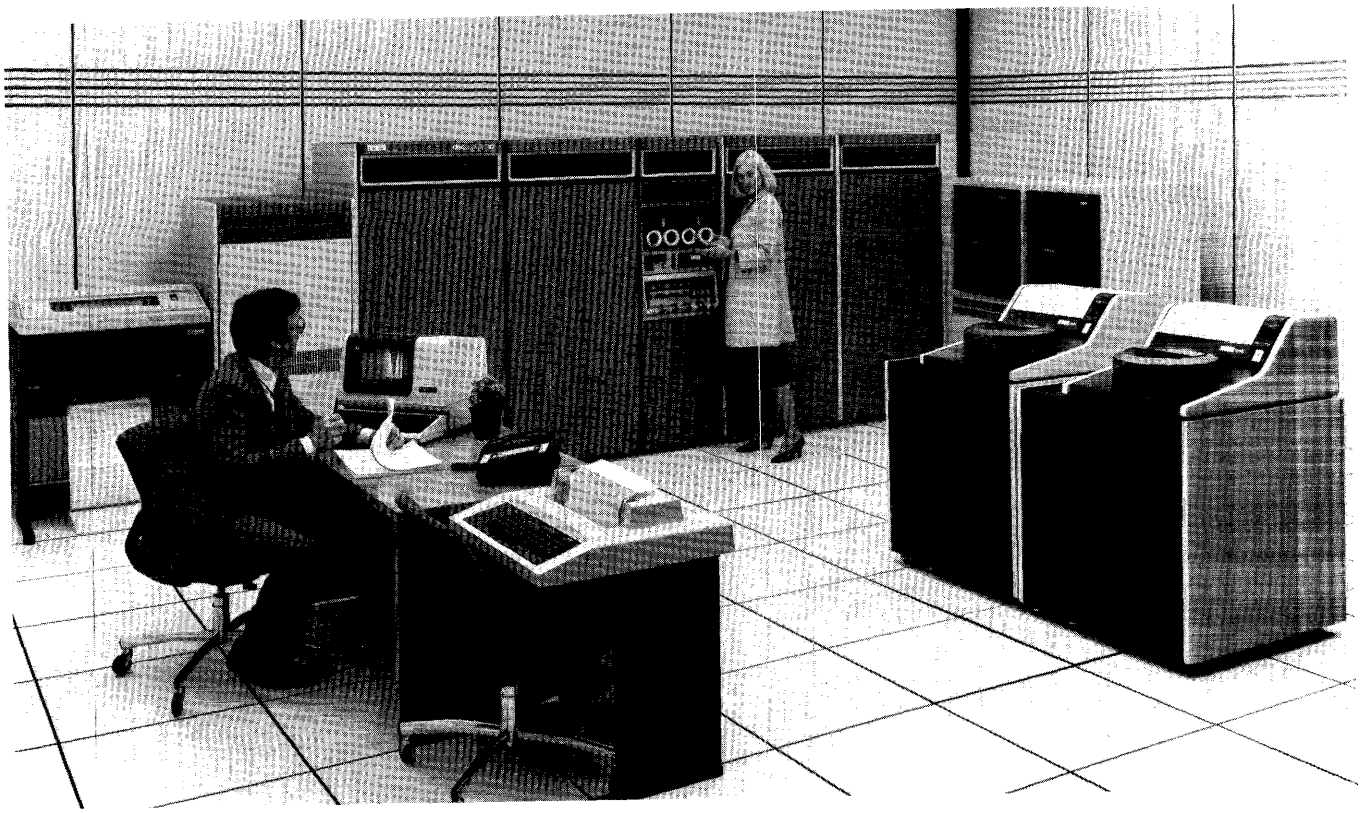
The CTY and remote diagnosis link can be operated in either user mode or console mode. Commands are the same in either local or remote CTY cases, except that the remote link repertoire is restricted. In user mode, the CTY is a user terminal, and commands are passed to and from the KS10 CPU under control of the console program. An exception is typing a CTRL \ which causes the console program to switch the CTY from user mode to console mode. All other commands performed in user mode are a function of the operating system.

In console mode, commands are directed to an executive by the microprocessor console hardware. An operator can perform the following features:

- Reset and bootstrap the system
- Load and check the KS10 microcode
- Deposit and examine memory
- Read and write I/O device registers
- Transmit to and receive from KS10 (backplane) bus
- Start and stop CPU clock
- Single-step the CPU clock
- Execute a given instruction
- Halt the machine
- Start the machine at a given location

The console program initializes the CTY to console mode at power up. When in console mode, either starting execution, continuing execution, or typing a CTRL Z will cause the CTY to switch to user mode. A CTRL \ in user mode causes a return to console mode. Also, an error which lights the fault indicator causes a return to console mode, as does any KS10 processor halt instruction.

7
The Peripherals



DECsystem-10s support a family of peripheral devices that includes disks, magnetic tapes, terminals and terminal interfaces, lineprinters, card readers, and a paper tape punch/reader.

The wide selection of peripheral devices reduces the storage burden on disks and provides convenient media for archiving files. Programs or data files that are infrequently used can be stored on magnetic tape which can be easily transported and accessed. TOPS–10 supports multiple card readers and paper-tape-punches/readers. Production of hard-copy output can be improved by using multiple lineprinters with different characteristics.

The terminals supported by the DECsystem-10 are the hard-copy LA37, LA38, LA120 and the VT100 video terminals.

A DN20 communications processor that support synchronous communications is available for DECsystem-10s.

The DN200 remote station is discussed under the heading Remote Job Entry Stations in Section 10, Communications.

COMPONENTS

DECsystem-10s support four types of peripheral systems:

- Mass storage peripherals — disks and magnetic tapes
- Unit-record peripherals — lineprinters, card readers and paper tape devices
- Terminals and terminal line interfaces
- Communication front-end processors

Processor I/O Subsystems

The KS10 (DECSYSTEM–2020) processor's peripheral devices are UNIBUS and MASSBUS devices (see Section 6) that interface to the system through UNIBUS adapters (UBAs). The UBA is a single extended hex module connecting to both the KS10 backplane bus and a UNIBUS. Two UBAs are standard in the DECSYSTEM–2020. One UBA and UNIBUS are reserved for disks. The second UBA and UNIBUS are used for all other devices: tape, lineprinter, and synchronous and asynchronous communications lines.

Details of the KL10–E I/O subsystem can be found in Section 5 under Input/Output Subsystem.

Mass Storage Peripherals

The DECsystem-10 mass storage peripherals are moving-head disk drives and magnetic tape transports.

Disks

The disk subsystems provide high performance and reliability. Disk features include accurate servo positioning, error correction, and offset positioning recovery. Table 7–1 summarizes the capacities and speeds of the disk devices.

To support the performance and reliability features of the system's disk devices, the operating system's disk device drivers provide comprehensive error recovery algorithms (for example, ECC and offset recovery for disk) and log all device errors.

Disk controllers allow several drives to perform simultaneous seek operations. Since the controller is not busy during seek operations, data transfers on one drive can overlap seeks in progress on other drives. If more than one drive is on the same controller, overlapped seeks will accelerate processing for the disks on that controller. Overlapped seeks increase throughput by decreasing the effective access time.

RM03 Disk Pack Subsystem (KS system only)

The RM03 is a top-loading, free-standing disk drive housed in a dedicated cabinet. Subsystems are expandable to eight disk drives or 536 Mb.

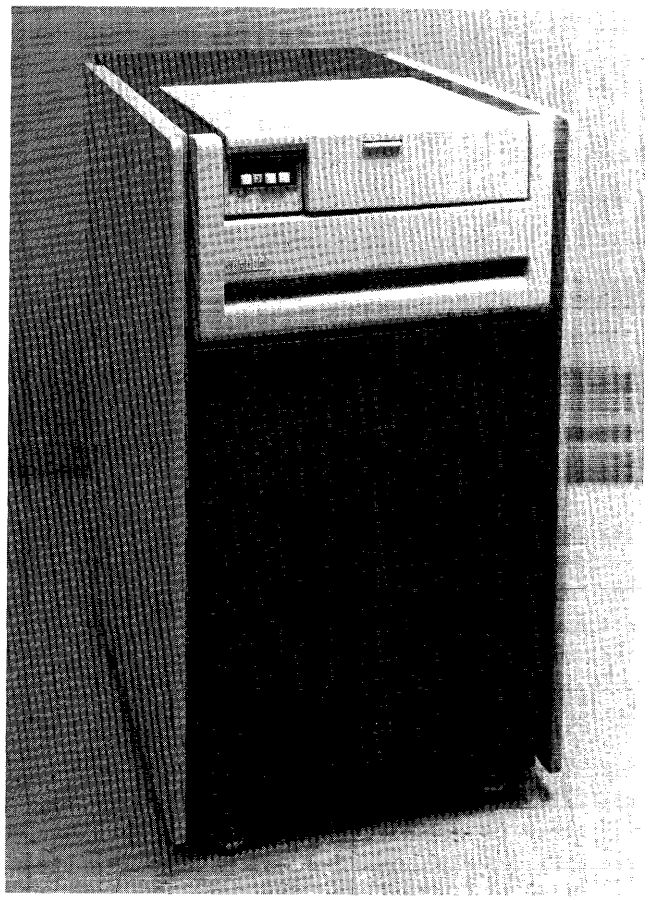


Table 7–1
Disk Devices

Disk	Type	Capacity (Mbyte)	Peak Transfer Rate (Kbytes per second)	Average Seek Time (ms)	Average Rotational Latency (ms)	Interface	Maximum Drives per Controller
RM03	Removable	67 Mb	1200	30	8.30	MASSBUS	8
RP06	Removable	176 Mb	806	28	8.33	MASSBUS	8
RP20	Non-removable	967 Mb	1200	25	8.33	MASSBUS	4*

*Two spindles per drive
K = 1,024; M = 1,000,000

The RM03 has a formatted capacity of 67 megabytes and an average access time of 38 milliseconds. Its maximum data transfer rate is 250,000 36-bit words per second. The RM03 disk does 18-bit NPR data transfers over a UNIBUS and 36-bit NPR data transfers over the KS10 backplane bus (see Section 6).

RP06 Disk Pack Subsystem (KS and KL systems)
The RP06 is a large capacity disk drive. This disk offers both high performance and a low price per megabyte of storage. It uses a removable disk pack and has a wide variety of large disk features.

RM03 and RP06 disk drives can be mixed on the same RH11 controller on a KS10; however, they can not be mixed within the same logical file structure.

RTP20 Disk Subsystem (KL systems only)
The RTP20 is the highest capacity disk offered by DIGITAL. The RTP20 consists of an RP20 disk drive and controller. The RP20 is a dual-spindle device with a formatted capacity of 967 megabytes. Master Subsystems are packaged to include an RH20/DX20 Channel, RTP20 Controller, and RP20 Master Disk. When additional capacity is required, the susbsystem can be expanded to include up to three add-on drives (1200 megabytes unformatted).

Each RP20 data module has fifteen recording surfaces with two read/write heads per surface and a transfer rate of 1.2 megabytes per second with a 25 ms average access time. Each data module may transfer data independently with the optional dual-port feature.

Tape Devices
DECSYSTEM–2020s are available with TU77 magnetic tape drives. Other DECsystem-10s are available with this drive plus the TU70 series tape subsystems. These provide capacities and speeds needed for the wide range of DECsystem-10 applications.

All tape devices use a mylar-base, oxide-coated magnetic tape with reflecting marker strips to indicate Beginning of Tape and End of Tape. Adjacent files are separated by formatted interrecord gaps. In addition, the use of industry-compatible formats facilitates data

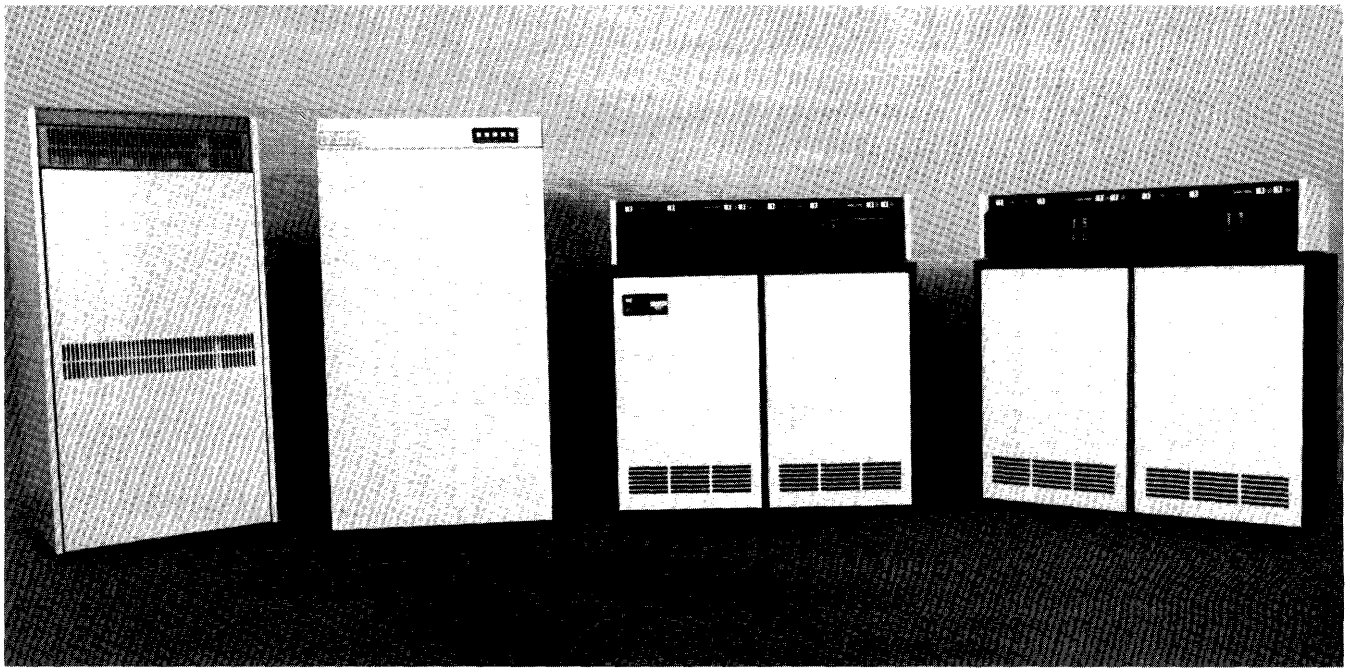


Table 7–2
Tape Devices

Tape	Type	Size	Capacity (Mbytes)	Column Buffering	Recording Density (p1)	Read-Write Speed (ips)	Max. Data Transfer Rate (K char/sec)	Rewind Speed (ips)	Interface	Maximum per Controller
TU72	9-track	10.5″ reel	40	Vacuum	1600, 6250	125	750	500	MASSBUS	8
TU77	9-track	10.5″ reel	40	Vacuum	800, 1600*	125	200	440	MASSBUS	4

* Program-selectable
K = 1,024; M = 1,000,000

transfer among computers and reduces hardware costs.

Several common features ensure data integrity and reliability, optimize performance, and facilitate maintenance.

All tape devices provide a write-protect capability to protect the integrity of data that is read and written onto the tape. An industry-standard write-protect ring is located on the tape reel. The tape drive can sense write-protection and prohibit data writing.

Accurate data recording and retrieval is ensured on all tape systems with read-after-write checks. This check verifies that proper data is written on the tape, thus eliminating the chance of data being written on worn or damaged sections of tape. If an error is detected in the read-after-write check, a message is sent to the processor.

Parity checks and longitudinal and cyclic redundancy checking further ensure the accurate transfer of data in all magnetic tape systems. Parity is checked character by character when reading and writing on tape at 6250(GCR) and 1600(PE) bits-per-inch. 800(NRZI) bits-per-inch operation also includes a cyclic redundancy check (CRC) character and longitudinal redundancy check (LRC) character. CRC and LRC characters are calculated when a block is written and checked when the block is read. If an error is detected, an indication is made to the host computer.

All magnetic tape systems minimize bad-tape-error problems through a runaway timer that allows the system to recover from bad tape sections on the reel.

Magnetic tape controller/formatters all perform similar operations. These include:

- Moving the tape to new positions in forward or reverse
- Monitoring tape operation
- Fetching, formatting, and sending data
- Handling error conditions and drive servicing requirements

TU72 Series Magnetic Tape System

The TU70 series tape subsystem consists of fully integrated, high performance magnetic tape storage systems that are specifically designed to operate on a DECsystem-10. The TU70 series tape drives provide high quality handling of ANSI standard 12.7 mm magnetic tape at up to 750,000 characters per second. The subsystem consists of one or more TU72–E (9-track, 125 inch per second, and recording densities of 1600 and 6250 bits-per-inch) tape drives, and a maximum of eight drives, a TX02 tape control unit, and a DX20 data channel. With the TU72, the tape control unit connects to each drive with radial cabling, and the DX20 data channel connects directly to the internal data channel of the DECsystem-10. In addition, two optional switches are available, the TX03 and TX05. The TX03 allows the user to interface between two data channels, one control unit and up to eight drives. The TX05 allows interface operations between two data channels, two control units and up to sixteen drives.

When reading and writing in PE mode, the accuracy of data transfer is confirmed with character-by-character parity checking. Error correction for single-track dropout is made on-the-fly. The GCR mode uses a polynomial error detection scheme on data blocks within each record. Data reliability is increased with a self-clocking feature that is independent of tape skew.

TU77 Magnetic Tape System

The TU77 is a high-performance tape storage system that is also suitable for many high duty-cycle applications, such as disk-to-tape backup and transaction processing. Design considerations, such as the elimination of mechanical relays and incandescent lamps on the drive and the use of air bearings and ceramic guides for reduced media wear, ensure its reliability. A tape interlock disables tape motion if there is a pressure loss in the vacuum system. This feature reduces the chance of accidental tape damage.

The TU77 is a fully integrated system that is packaged with its associated interface and power supply in a dedicated cabinet. The TU77 subsystem consists of a

TU77 tape drive, a dual-density TM03 tape formatter, and a MASSBUS controller that mounts in the processor chassis. Each subsystem can include up to four tape drives with a total maximum of 16 drives per DECsystem-10.

Automatic tape threading maximizes operator convenience and minimizes tape handling. Smaller reels of tape can be threaded automatically when tape is placed manually in the loading slot. If a tape fails to load on the first attempt, the TU77 will detect the misfeed and reload without operator intervention (10.5″ reels only).

The TU77 subsystem also provides data integrity through automatic correction of single-track errors. In PE mode, this error correction is done automatically by the hardware. In NRZI mode, error correction is performed under software control.

The TU77 reads in both forward and reverse directions and uses the industry-standard data format.

Unit-Record Peripherals

The DECsystem-10 supports a full line of unit-record peripherals including lineprinters and card and paper-tape devices.

LP20–A and –B Lineprinters

The LP20–A and –B lineprinter systems have two hardware components: an LP05 lineprinter and an LP20 lineprinter controller and data source interface.

The LP05 lineprinter is a medium duty-cycle drum line printer. It is designed for data processing environments that require good print quality and medium print volumes. The LP05 is an impact-shaped (whole) character, 132-column lineprinter. It prints 300 lines per minute using a 64-character set, or 230 lines per minute using a 96-character set. It performs a single-forms (paper) step in 45 milliseconds (maximum) and, when formatting vertically, slews forms at up to 67.8 cm per second.

Up to 132 characters can be stored in a print line buffer and, upon command from the data source, the LP05 prints the contents of the buffer and advances the forms as specified by the command. It signals the data source when it is ready for the next line of print data or forms motion command.

The LP05 uses a rotating drum containing all the characters in front of the forms and ribbons. Fifty-eight print hammers behind the forms are time-shared between 132 data columns to produce inked characters and carbon transfer characters on multicopy forms.

The LP05 contains a 12-channel programmable vertical format unit (PVFU). It uses a format memory that is loaded from the data source, so no format tape is installed by the operator and there is no risk of running a job with the wrong format tape. The PVFU can be loaded any time data is requested by the printer. Format memory data and control codes are transmitted to the printer via the normal data lines using the standard demand/strobe communications. The PVFU can control the vertical movement of forms having eight to 143 lines.

The LP05 lineprinter has a panel with switches and indicators for operation and operator-level maintenance. The printer can be dynamically tested with its

self-test module. When the print TEST/ON–LINE switch is set to TEST, the self-test module operates as a built-in data source; the printer communicates with the self-test module on a demand/strobe basis, stores the data received in the line buffers and paper motion registers, as appropriate, and processes the data. After the line of data has been printed and the paper moved, the printer resumes the data exchange communication with the self-test module in a continuous cycle.

A long line interface provides the LP05 with differential receivers and drivers so that it can be located up to 30.5 meters from the data source. A standard 7.6 meter device cable is supplied when specified.

LP20–C and –D Lineprinters

The LP20–C and LP20–D consist of two hardware components: an LP14 lineprinter and an LP20 line printer controller and data source interface.

The LP14 lineprinter is a medium-duty cycle drum lineprinter. It is designed for use in data processing environments that require good print quality and medium print volumes. The LP14 is an impact type, shaped (whole) character, 132-column lineprinter. It prints 890 lines per minute using a 64-character set or 650 lines per minute using a 96-character set. It performs a single-forms (paper) step in 20ms and slews forms at up to 76.2cm (30 inches) per second when formatting vertically.

The LP14 stores a stream of up to 132 characters in a print line buffer and, upon command from the data source, prints the contents of the buffer and advances the forms as specified by the command. It signals the data source when it is ready for the next line of print data or forms motion command.

The LP14 uses a rotating drum containing all characters in front of the forms and ribbon and 132 print hammers, which are behind the forms, to produce inked characters on the front of the forms and carbon transfer characters on internal pages if multiple-copy forms are being used.

The LP14 contains a 12-channel Direct Access Vertical Format Unit (DAVFU). The DAVFU uses a format memory that is loaded from the data source. This relieves the operator from having to install a format tape and eliminates the risk of running a print job with the incorrect format. The DAVFU may be loaded any time data is requested by the printer. Format memory data and control codes are transmitted to the printer through the normal data lines using the standard Demand/Strobe communications. The DAVFU can control the vertical movement of forms having a minimum of 8 lines and a maximum of 143 lines.

The LP14 comprises an operation/maintenance panel that mounts switches and indicators to operate the line printer and perform operator-level maintenance. A self-test module allows the LP14 to be tested under dynamic conditions. When the printer's TEST/ON–LINE switch is set to test, the self-test module operates as a built-in data source; that is, the printer communicates with the self-test module on a Demand/Strobe basis, stores the data received in the line buffers and paper motion registers as appropriate, and processes the data. After the line of data has been printed and paper moved, the printer resumes the data exchange communication with the self-test module in a continuos cycle.

LP100 Lineprinters (1090 systems only)

The LP100 lineprinter series currently offers three option type variations. The LP100–B consist of an LP07 lineprinter and a LP100 controller; the LP100–D/E consist of an LP05 lineprinter and a LP100 controller; the LP100–F/H consist of an LP14 lineprinter and an LP100 controller. For a description of the LP05 and LP14 lineprinters read the two previous sections. For a description of the LP07, read the following section.

LP200 Lineprinters (1091 systems only)

The LP200 lineprinter system has two hardware components: an LP07 lineprinter and an LP20 lineprinter controller and data source interface.

The LP07 is a high-performance horizontal font motion lineprinter. It is designed for data processing environments that require high-grade print quality, heavy print volumes, and high reliability.

The LP07 is an impact type, shaped (whole) character 132-column lineprinter. It prints 990 or 1220 lines per minute using a 64-character set and 715 or 905 lines per minute with a 96-character set. Print speed is operator selectable. The LP07 does a single-forms paper step in 12.5 milliseconds and slews forms at up to 152.4 cm (60 in.) per second when formatting vertically.

Up to 132 characters can be stored in a print line buffer and, upon command from the data source, the LP07 prints the contents of the buffer and advances the forms as specified by the command. It signals the data source when it is ready for the next line of print data or forms motion command.

The LP07 uses a Charaband (Charaband is a trademark of Dataproducts Corporation) as the horizontal font carries in front of the forms. The 132 print hammers behind the forms and the ribbon produce the inked characters and carbon transfer characters on multicopy forms. The Charaband has the advantages of train printers but minimizes the problems of character set rigidity, friction, and wear associated with other horizontal font techniques.

The Charaband has two complete character sets, one on each side. A one-minute manual operation switches the sets.

The LP07 lineprinter contains a direct-access vertical form unit (DAVFU) that provides the same benefits as the PVFU discussed in the LP20 section, above. The DAVFU also permits six or eight lines/inch print density under program control.

The LP07 contains a self-test unit similar in function to the unit discussed in the LP20 section, above.

A long line interface provides the LP05 with differential receivers and drivers so that it can be located up to 152 meters from the data source. A standard 7.6 meter device cable is normally supplied.

Card Readers

The CD20 card readers interpret encoded, punched information using the American National Standard 8-bit card code and use a special punch outside the data representation to indicate end-of-file. The card readers use the industry-standard EIA card that has 80 columns and 12 zones, or rows.

The card readers are designed to meet different throughput requirements. CD20 card reader is available in both table (CD20–A) and console (CD20–C) models. CD20–A can process 285 punched cards per minute, and the CD20–C, 1200 cards per minute.

The CD20–A table top unit has an input hopper capacity of 1000 cards. The CD20–C has a capacity of 2250 cards.

Card reader design helps prevent card jams and keeps card wear to a minimum. Readers also have a high tolerance to cards that have been nicked, warped, bent, or subjected to high humidity. Readers use a short card path, with only one card in the track at a time. They all use a vacuum pick mechanism that keeps cards from sticking together by blowing a stream of air through the bottom half-inch of cards in the input hopper.

The CD20–C console card reader has a single piece read station with infrared light-emitting diode emitters and phototransistor detectors. No adjustments are required in the ten-year life expectancy of the diodes.

The console model also provides high data integrity with a "resync/error detection" feature. The data strobe can be resynchronized for each data column. The reader will either correctly read a misregistered card or reject the card by halting with a "read check" indication.

Cards can be loaded or unloaded while the console model is operating. A switch can be set to provide either blower shutdown or continued running after the last card has been read. Automatic shutdown reduces computer room noise levels and can also signal the operator that the card hopper is empty.

PC10/20 Paper Tape Reader/Punch

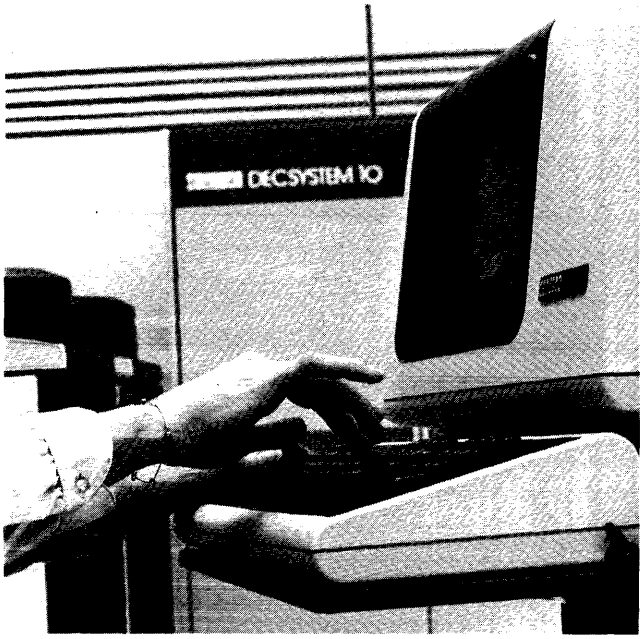
The PC10/20 (PC10 is used on 1090 systems and the PC20 is used on 1091 systems) reads eight-channel paper tape at 300 lines per second and punches 50 lines per second in either alphanumeric or binary. It automatically fan-folds the paper tape.

The PC10/20 contains a photoelectric paper tape reader and an electromechanical punch. A set of photodiodes translates the presence or absence of holes in the tape to the logic level of 1s and 0s.

The device operates in either alphanumeric or binary mode. In alphanumeric mode, a single tape-moving command reads all eight channels from the first line encountered. In binary mode, the device reads six channels from the first six lines in which hole 8 is punched and assembles the information into a 36-bit word. The PC10/20 interface contains a 36-word buffer from which all data is retrieved by the processor.

At 300 lines per second the reader takes 3.33 milliseconds per alphanumeric character and 20 milliseconds per contiguous binary character.

The punch perforates up to 50 lines of tape per second in alphanumeric or binary mode.



Terminals and Interfaces

A KS system can handle a maximum of 32 local and remote line interfaces. A KL system can handle up to

512 local and remote lines. Programs can control terminal operations through the terminal driver. The terminal driver receives and services interrupts, initiates I/O operations, cancels in-progress I/O operations, and performs other device-specific functions. A program can perform the following functions:

- Get data from an open terminal without stalling the program execution. The program does not have to wait for incoming data to be available in the terminal input buffer.
- Through echo control mode, a full-duplex terminal can be used to simulate the operations of a block-mode terminal. Input data from the terminal is processed a field at a time, without affecting the displayed output in other fields on the screen.
- Translate, interpret, and transmit ESCAPE sequences.

In addition, pseudoterminals can be used for jobs that do not require operator intervention (such as batch jobs). A pseudoterminal is a nonphysical device that has the characteristics of a terminal but has no physical device attached with it. Like a terminal, a pseudoterminal has both input and output buffers to which user programs send input and from which they extract output. By using pseudoterminals, one job can control other jobs on the system.

Terminal characteristics are initially established in a command file during software installation. However, system users also can modify the characteristics of their particular terminal. A few of the settable terminal characteristics users can modify are:

- Set the width of the print line between 1 and 254 columns. As a result, the system automatically generates a carriage return and line feed sequence after the specified number of characters have been typed or printed.
- Control transmission of uppercase and lowercase characters.
- Allow the computer to interrupt transmission of characters from the terminal (XOFF) or instruct the terminal to resume transmission of characters (XON). The terminal hardware must be present to respond to XOFF and XON characters.
- Set the rate at which the terminal's interface can accept or pass characters.
- Set even, odd, or no parity checking.

A line entered on a command terminal is terminated by any of several special characters. The RETURN key, for example, is one of the most common means of transmitting a command to the host. A program reading from a terminal can optionally specify a particular line terminator for read requests.

LA120 Hard-Copy Terminal

The LA120 is a hard copy terminal with high throughput and a number of advanced keyboard-selectable formatting and communication features. It uses a contoured, typewriter-style keyboard and includes an additional numeric keypad and a LED display for terminal characteristics.

Several features give the LA120 its high throughput:

- 180-character-per-second print speed
- 14 data transmission speeds ranging up to 9600 baud
- 1024-character buffer to equalize differences between transmission speeds and print speeds
- Smart and bidirectional printing, so that printhead always takes shortest path to next print position
- High-speed horizontal and vertical skipping over white space

In addition to its throughput, the LA120 is distinguished by its printing features. The terminal offers eight font sizes, ranging from expanded (five characters per inch (cpi)) to compressed (16.5 characters per inch). Hence a user could, for example, select a font size of 16.5 cpi and print 132 columns on an 8/4-inch-wide sheet. Other print features include six line spacings ranging from 2 to 12 lines per inch, user-selectable form lengths up to 14 inches, left/right and top/bottom margins, and horizontal and vertical tabs.

The LA120 is designed for easy use. Terminal characteristics are selected by clearly labelled keys and simple mnemonic commands. Once the selections have been made, the operator can check his settings by depressing the STATUS key. The terminal will then print a listing of the selected settings.

LA38 Hard-Copy Terminal

The LA38 DECwriter IV is a low cost, desktop, microprocessor-driven terminal capable of processing data at a rate of up to 30 characters per second. The LA38 includes a universal power supply, a standard EIA interface and EIA null modem cable, an 18-key numeric keypad, paper-out switch, paper-feed tractor, and user-assistance documentation package.

Features such as horizontal tabs, horizontal margins, and a choice of four character sizes and six line spacings can be set and changed by the host computer or by the user. The LA38 has a permanently stored format for a computer printout. When the terminal is powered up it automatically assumes a 10 characters per inch character pitch, 6 lines per inch vertical spacing, tab stops every 8 spaces, left margin set at column 1, and right margin at column 132.

The LA38 operates at 300 baud and can print at burst speeds up to 45 characters per second. An alternate speed of 110 baud and 10 characters per second can be selected from the keyboard. The desktop configuration and sculptured typewriter-like keyboard are so similar to standard typewriters that the transition from typewriter to terminal is natural.

The terminal's basic design contributes to its reliability and maintainability. A single logic/power amplifier board with custom LSI reduces the component count and increases circuit reliability.

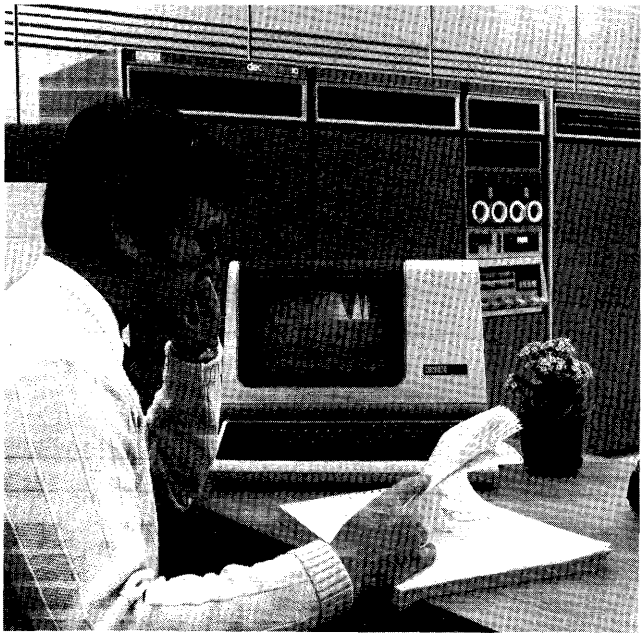
The LA38 senses printhead jams instantly. When a printhead jam occurs the power is removed from the printhead drive until the jam is corrected and the terminal restarted. This action prevents motor overloads and blown fuses. The highly reliable printhead has been designed and tested to print over 100 million characters. The printhead can be adjusted to adapt to various forms thicknesses. A paper-out sensor generates a signal if there is no longer any paper in the underside slot. Users may define any of four actions to be taken if a paper fault condition occurs; or, if desired, the paper-out sensor may be disabled.

Although the LA38 has been designed to operate reliably without scheduled preventive maintenance, if a problem occurs, the unit disassembles simply and quickly for easy access to all components. Printing self-test diagnostics allows quick and accurate identification of any faulty components. In addition, with the new snap-in style ribbon cartridge, users can change ribbons quickly and easily.

Wherever possible, ANSI standard escape sequences are used. These same escape sequences are also implemented on DIGITAL's LA120 and VT100, ensuring compatibility among DIGITAL's terminal products.

VT100 VIDEO TERMINAL

The VT100 video terminal is an uppercase/lowercase ASCII terminal that offers a variety of user-controllable character and screen attributes. The VT100 features a typewriter-like detachable keyboard that includes a standard numeric/function keypad for data entry applications. Also featured are seven LEDs, four of which are program-controlled, that can be used as operator information and diagnostic aids.



The VT100 offers a number of advanced features.The most important of these are:

- Ability to select either of two screen sizes: 24 lines by 80 columns or 14 lines by 132 columns
- Ability of programs to select on a line-by-line basis either double-width/single-height characters or double-width/double-height characters
- Smooth scrolling and split screen capability
- Ability to set baud rates, tabs, and Answer Back messages from the keyboard and to store these in RAM (Random Access Memory)
- Special line drawing graphic characters which provide the ability to display simple graphics for business or laboratory applications
- Ability to select black-on-white characters or white-on-black characters on a full screen basis

In addition, several options further extend the capabilities of the VT100. These include the advanced video option that adds selectable blinking, underline, and dual-intensity characters to the existing reverse video attribute, and additional RAM allowing 24 lines of 132 characters.

COMMUNICATION HARDWARE

The following sections describe the communication hardware supported by DIGITAL.

DZ11 Terminal Line Interface (KS and DN25 systems)

The DZ11 is a serial line multiplexer whose character formats and operating speeds are programmable on a per line basis. A DZ11 connects the UNIBUS with a maximum of 8 asynchronous serial lines. Each line can run at any of 15 speeds.

Local operation with EIA terminals is possible at speeds up to 9600 baud. The DZ11 may be used with dial-up, full-duplex terminals that operate through most industry-standard modems that run at 300 or 1200 bits per second. The DZ11 optionally generates parity on output and checks parity on input. Incoming characters are buffered using a 64-character silo buffer. Outgoing characters are processed on a programmed interrupt request basis.

As many as 16 DH11s may be placed on a single processor, creating a total capacity of 128 lines.

DL11 Serial Line Asynchronous Interfaces

The DL11 is an interface between a single, asynchronous, serial communication channel and the processor. It performs serial-to-parallel and parallel-to-serial conversion of serial start/stop data with a double-character buffered MOS/LSI circuit called a UART (Universal Asynchronous Receiver-Transmitter). This 40-pin, dual in-line package includes all of the circuitry necessary to double-buffer characters in and out, serialize/deserialize data, provide selection of character length and stop code configuration, and present status information about the unit and each character.

With a DL11 interface, a DECsystem-10 can communicate with a local terminal such as a console teleprinter, with a remote terminal via data sets and private line or public switched telephone facilities.

Users can order the data rate from a selection of 13 standard rates up to 9600 bits per second, or they can order a nonstandard rate device. With most of the standard rates, the interface can offer split-speed operation for faster, more efficient handling of computer output. In addition, character size is strap- or switch-selectable, and parity checking (even, odd, or none) and stop code length (1, 1.5, or 2 bits) are selectable.

DUP11 Single Synchronous Line Interface (KS systems only)

The DUP11 is a character-buffered, synchronous serial-line interface capable of two-way simultaneous communications. The DUP11 translates between serial data and parallel data. Output characters are transferred in parallel from the PDP–11 UNIBUS into the DUP11, where they are serially shifted to the communication line. Input characters from the resident modem are shifted into the DUP11 and made available to the processor on an interrupt basis.

The self-contained unit is capable of handling a wide variety of protocols including byte-oriented protocols such as DDCMP and BSC, and bit-oriented protocols such as SDLC, HDLC and ADCCP. Signals needed to establish communications with the Bell Series 200 synchronous modems are resident in the DUP11's Receive Status Register. The DUP11 can transmit data at up to 9600 bits per second (limited by modem and data set interface level converters).

Its capabilities make the DUP11 well suited for remote batch, remote data collection, remote concentration, and network applications. In addition, multiple DUP11s can be used in applications requiring several synchronous lines.

KMC11–A AUXILIARY PROCESSOR (KS system only)

The KMC11–A is an auxiliary processor, complete with memory, that interfaces to the UNIBUS. The KMC11 improves the performance of the DECsystem-10 computer systems by performing time-consuming system functions in parallel with the host CPU, thereby off-loading it. It is especially suited to controlling I/O operations, such as data communications and analog I/O, that require extensive intelligence.

The KMC11 is a high-speed MSI microprocessor (300 nsec instruction time), uses 16-bit microinstructions, and operates on 8-bit data paths. Its 1024 16-bit word writable control memory contains the microprogram and is loaded by the host processor. A 1024 8-bit byte data memory stores frequently used information for high-speed access by the microprogram. NPR UNIBUS interface provides access to control, status, and data registers of one or more peripherals on the UNIBUS. This enables low cost programmed-I/O devices to operate as if they had an intelligent DMA capability.

An external connector allows the KMC11 to be connected directly to a high-speed peripheral such as a DMC11 synchronous line unit. The full-duplex 8-bit parallel interface is well suited to custom designed interfacing.

The KMC11 is complete on a single printed circuit module and includesr the microprocessor, control memory, data memory, UNIBUS interface, and external connector.

DN20 Communications Front End

The DN20 is designed to handle all network communications functions. The DN20 is referred to as a communication front-end in order to distinguish it from the console front-end that controls the local command terminals and unit-record peripherals. The DN20 and console front-end processor communicate with the KL processor through the DTE hardware interface. The DTE resolves the differences in word size between the KL (36-bit words) and the DN20 (16-bit words). In the case of the KS10 processor, there is no separate communications front-end and the communications software resides entirely in the KS10 processor.

DN200 Remote Station

The DN200 Remote Station provides remote job entry and/or remote concentration of terminal data for high-speed sychronous transmission in DECsystem-10 networks. Features include down-line loading of system software and diagnostics, easy expandability from a remote job entry terminal or remote concentrator to a fully expanded remote station, support capabilities of up to 32 asychronous lines, 2 sychronous lines, a card reader, and a line printer.

8
The Languages

In addition to assembly language (MACRO), the TOPS–10 system software supports many optional high-level programming languages: FORTRAN, COBOL, ALGOL, BASIC, APL, BLISS, and CPL. This variety permits the user to use the most effective or familiar programming solution to match the problem.

FORTRAN–10 is a globally optimizing compiler and a run-time system with an interactive debugger. Both have been designed to simplify the programmer's job, provide superior compile-time and run-time diagnostics, facilitate debugging, and produce error-free and fast running programs.

COBOL–68 and COBOL–74 are a high-level language implementation designed specifically for business data processing. COBOL can be used to create on-line terminal applications or to write batch applications.

Although BASIC is an ideal language for novice programmers who need a fast, easy way to solve problems, it is also a powerful and efficient language suitable for sophisticated applications. BASIC is a compiler that produces fast-running programs, yet it retains the highly interactive immediate mode feature of primitive BASICs.

ALGOL–10 is a scientific language designed for describing computation processing (algorithms). It is a problem-solving language in which the problem is expressed as complete and precise statements of a procedure.

Two levels of the popular APL language are available. Both levels are full APL implementations with significant extensions. The basic version suits users who do not require the file I/O or the advanced APL function. The extended version of APL, APL–SF, has all of the features the basic version has, plus advanced features that substantially increase the range of applications for which it can be used.

BLISS–36 is DIGITAL's implementation language for software development. It contains many of the features of a modern high-level language, yet it also provides the flexibility and access to hardware of assembly language.

CPL (conversational programming language) is a PL/I-like interpreter. It is a well documented and easy-to-learn subset of ANSI–1976 PL/I.

TOPS–10 Assembler

MACRO is the DECsystem-10 symbolic assembly language. It makes machine language programming easier and faster for the user by:

- Translating symbolic operation codes in the source program into the binary codes needed in machine language instructions
- Relating symbols specified by the user to stored addresses or numeric values
- Assigning relative memory addresses to symbolic addresses of program instructions and data
- Providing a sequentially numbered listing with symbols cross-referenced to show where they are defined and where they are used

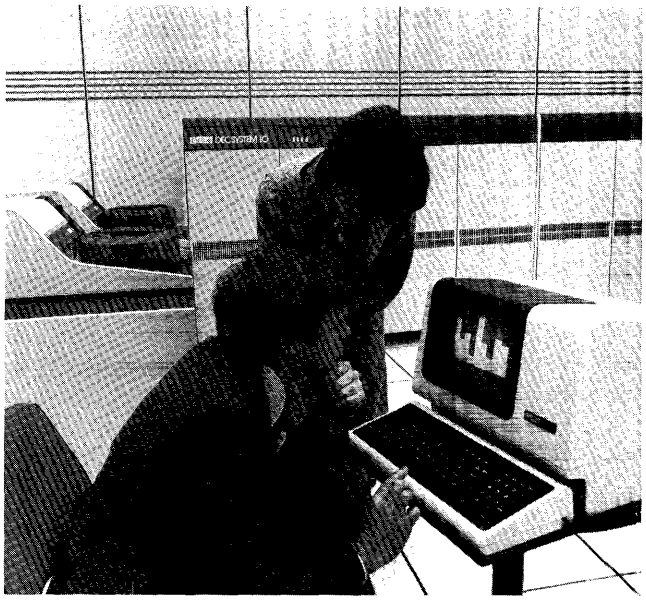
MACRO programs consist of a series of free format statements that can be prepared on the user's terminal with one of the system's editing programs. The elements in each statement can be entered in free format. The assembler interprets and processes these statements, generates binary instructions or data words, and produces a listing which can contain cross-reference symbols for ease in debugging the program. MACRO is a device-independent program; it allows the user to select, at run time, standard peripheral devices for input and output files. For example, input of the source program can come from the user's terminal, and output of the program listing can go to the lineprinter. More commonly, the source program input and binary output are disk files.

The MACRO assembler contains powerful macro capabilities that allow the user to create new language elements. This is useful when a sequence of code is used several times with only certain arguments changed. The code sequence is defined with dummy arguments as a macro instruction. Thus, a single statement in the source program referring to the macro by name, along with a list of the real arguments, generates the entire sequence needed. This capability permits the expansion and adaptation of the assembler in order to perform specialized functions for each programming job. In addition, by changing just the macro definition, the programmer changes the definition for every call.

FORTRAN

FORTRAN–10 is a globally optimizing compiler and a run-time system with an interactive debugger. Both have been designed to simplify the programmer's job, provide superior compile-time and run-time diagnostics, facilitate debugging, and produce error-free and fast running programs.

The FORTRAN language is based on the 1966 American National Standard FORTRAN Language and has many extensions to that standard. It also includes many features from the ANSI 1977 standard FORTRAN. Both the compiler and object-time system are reentrant (sharable). FORTRAN features include:



- Accepted and powerful features for handling a wide range of technically-oriented users
- Computational features matched by full-scale data handling and data management facilities
- Global optimization
- PARAMETER statements providing symbolic specification of compile-time constraints
- INCLUDE statements allowing users to include in the compilation of a given program unit source code that resides in a file other than the primary source file
- n-dimensional arrays
- Ability to generate array bounds checking
- Direct-access I/O capabilities
- FORTRAN debugger that permits examination and modification of program data, statement-by-statement program tracing, and setting pauses on any statement or routine
- OPEN and CLOSE statements for file specification and control
- ENCODE/DECODE statements
- Boolean operators, including equivalence and exclusive OR, in addition to OR, AND, and NOT
- NAMELIST statement and list-directed I/O that provide format-free input and output operations
- Implied DO loops allowed in I/O and data statements
- Full mixed-mode arithmetic in expressions

- Octal constants
- Full-word masking operations allowed for all logic functions (rather than a result of just true or false)
- Relational operators
- Error-handling capabilities in I/O statements
- Device independence
- Multistatement lines
- Remarks in statement fields

The FORTRAN object-time system (FOROTS) controls the input/output, format interpretation, and numerical conversion for programs compiled by the FORTRAN compiler. The FORTRAN user can reference any mass storage, unit record, or terminal device. All special editing, conversion, and file structuring tasks are handled by the object-time system. Devices are normally specified by logical assignment so that physical device selection need not be made until run time. The devices corresponding to the specific I/O statements READ, PRINT, PUNCH, ACCEPT, and TYPE are also assignable at run time.

Language Extensions

Powerful FORTRAN extensions simplify program coding. Some of the enhancements are:

- Array subscripts — any arithmetic expression can be used as an array subscript. If the value of the expression is not an integer, it is converted to integer type.
- Alphanumeric literals — strings of characters bounded by apostrophes can be used in place of Hollerith constants.
- Mixed-mode expressions — mixed-mode expressions can contain any data type, including complex and byte.
- End of line comments — any FORTRAN statement can be followed on the same line by a comment that begins with an exclamation point.
- Read/write end-of-file or error condition transfer — the specifications END = n and ERR = n (where n is a statement number) can be included in any READ or WRITE statement to transfer control to the specified statement upon detection of an end-of-file or error condition. The ERR = n option is also permitted in the ENCODE and DECODE statements, allowing program control of data format errors.
- General expression DO and GO TO parameters — general expressions are permitted for the initial value, increment, and limit parameters in the DO statement and as the control parameter in the computed GO TO statement.
- DO increment parameter — the value of the DO statement increment parameter can be negative.
- Optional statement label list — the statement label list in an assigned GO TO is optional.
- Default FORMAT widths — the FORTRAN IV programmer can specify input or output formatting by type and default width and precision values will be supplied.
- Additional I/O statements — these include file control and attribute definitions; list-directed (free format) I/O; device-oriented I/O; memory-to-memory formatting; and unformatted direct access I/O, which allows the FORTRAN programmer to read and write files written in any format.
- Logical operations on INTEGER data — the logical operators .AND., .OR., .NOT., .XOR., and .EQV. may be applied to integer data to perform bit masking and manipulation.

Optimization

The FORTRAN–10 compiler performs many optimizations during compilation. The purpose of the optimizer is to prepare a more efficient object program that produces the same results as the original unoptimized program, but takes significantly less execution time. The output of the lexical and syntactic analysis phase of the compiler is developed into an optimized source program equivalent, in results, to the original program. The optimized program is then processed by the standard compiler code generation phase.

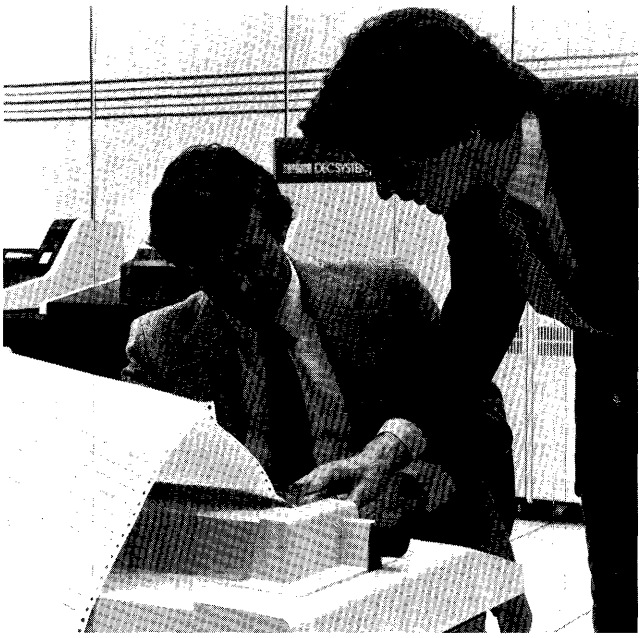
Debugging Tools

Two debugging facilities are available to the FORTRAN programmer: the FORTRAN Object-Time System and the use of a "D" in Column 1 of a FORTRAN statement.

The FORTRAN Object Time System (FOROTS) provides a traceback feature for fatal run-time errors. This feature locates the actual program unit and line number of a run-time error. Immediately following the error message, the error handler lists the line number and program unit name in which the error occurred. If the program unit is a subroutine or function subprogram, the error handler traces back to the calling program unit and displays the name of that program unit and the line number where the call occurred. This process continues until the calling sequence is traced back to a specific line number in the main program. The traceback feature lets the programmer determine the exact location of an error, even if it occurs in a deeply nested subroutine.

A "D" in Column 1 of a FORTRAN statement allows that statement to be conditionally compiled. These statements are considered comment lines by the compiler unless the appropriate debugging line switch is issued in the compiler command string.

Liberal use of the PAUSE statement and selective variable printing provide programmers with a method of monitoring program execution. This feature allows the inclusion of debugging aids that can be compiled in the early program testing stages and later eliminated without source program modification.



COBOL

COBOL–68 and COBOL–74 are optional high-level languages designed specifically for business data processing. COBOL can be used to create on-line terminal applications or to write batch applications. Under control of the multistream batch processor, program and data decks can be loaded into the card reader for spooling operations. However, using the same command language, the operation can be done from an interactive terminal.

COBOL features include:

- ANSI-standard compliance
- Quick, efficient program development
- Simple, interactive user application interface
- Efficient operation in both batch and on-line operation
- Programming tools with powerful yet easily used data editing, sorting, updating, and reporting features
- Remote access from on-line terminals or from remote stations
- Complete program and device independence for efficiency and reliability
- Ability to call subroutines written COBOL, FORTRAN, or MACRO

Listings produced by the compiler contain many documentation and debugging aids. English diagnostic messages are embedded in the source listing at the point of error. In addition, the listing can also include at the user's discretion a complete map of the object program and an easy-to-read listing of the compiled code. The latter is presented in the form used by the MACRO assembler. All object code is expanded to list the machine mnemonics and user-defined names in addition to the binary machine code (in octal). An implementation of the COBOL REPORT WRITER statement is provided.

Data Types

COBOL supports all the common COBOL data types, including packed decimal (COMP–3). COBOL supports the following data types:

- Numeric COMP–3, packed decimal data
- Numeric COMPUTATIONAL (COMP), binary data
- Numeric COMPUTATIONAL–1 (COMP–1), floating-point data
- Alphanumeric DISPLAY data (ASCII, EBCDIC, or SIXBIT)
- Numeric DISPLAY data (ASCII, EBCDIC, or SIXBIT)

These data types are required in a variety of applications and are provided for flexibility in specification and design.

String Manipulation

COBOL provides INSPECT, STRING, and UNSTRING verbs for character string handling. These verbs let programmers search for embedded character strings with TALLY and REPLACE, and they offer the ability to join or isolate separate strings with various delimiters.

Interactive COBOL Execution

The ACCEPT and DISPLAY statements of the PROCEDURE DIVISION allow easy terminal-oriented interaction between a COBOL program and the program user.

The program can receive user input lines with the ACCEPT statement. The ACCEPT statement can also retrieve the current date or time from the system.

The DISPLAY statement transfers data from a specified literal or data item to a specified device, normally the user's terminal. The statement can be modified by a special WITH NO ADVANCING phrase (without automatic appending of carriage return and line feed) that allows the COBOL program to control the format of the message sent. The WITH NO ADVANCING phrase causes the device to remain positioned on the

same line and the same character position following the last character displayed. This is especially useful when typing prompting messages on the terminal.

The ACCEPT and DISPLAY statements are intended primarily for use with keyboard devices. However, COBOL also allows the ACCEPT statement to accept cards from a card reader and the DISPLAY statement to display data on a line printer.

File Organization

The sequential I/O, relative I/O, and indexed I/O modules meet the ANSI–74 high-level standards (with the exception indicated in Table 8–1) and include all the COBOL verbs.

A complete index sequential-access mode (ISAM) package is included in the COBOL object-time system, allowing the user to access data either sequentially or randomly by key value on random-access devices. The time required to access a file is minimally affected by the number of additions made to the file. The technique of "chaining" records is not used. Instead, the index to the file is updated to minimize the number of accesses necessary to retrieve records.

LIBRARY Facility

With COBOL, programmers have a full ANSI–74 high-level library facility that includes high-level extensions (COPY...REPLACING). Frequently used data descriptions and program text sections can be held in library files that are available to all programs. These files can then be copied at compile time to reduce program preparation time and to eliminate a common source of errors.

CALL Facility

The CALL statement allows COBOL programs to invoke separately compiled subprograms, passing arguments in the process. These subprograms can be written in COBOL, FORTRAN, or MACRO. The CALL facility:

- Provides flexibility through modular development of application systems
- Permits functional separation of small, well-defined source modules
- Gives the programmer access to operating system-dependent features via subroutines written in MACRO

On-line Debugger

The on-line COBDDT debugging package permits user interaction during the execution of a program. No modifications are required to a source program . When the user wants to use COBDDT, the package is simply loaded with the object program when execution starts. The user can specify the points within the program at which to pause during execution. During these pauses, the user can examine and modify the contents of data items before proceeding. All references to data and procedure items are made by using the name in the source program. The user talks to the debugging package using familiar names rather than truncated or substituted names.

Source Program Input

The disk-resident compiler can accept source program input from all supported input devices, including input from source text library files stored on disks.

COBOL accepts source programs that are coded using either the conventional 80-column card reference format or the short, easy-to-enter terminal format.

- Terminal format is used with context editors controlled from an on-line terminal keyboard. It eliminates the line number and identification fields and allows horizontal tab characters and short lines. These capabilities offer potential savings in disk space and allow easier interactive input of source programs.
- Conventional format produces source programs that are compatible with the reference format of other COBOL compilers throughout the industry.

RERUN

The RERUN feature allows the user to periodically save the status of a job. In the event of a later disruption the job can be restarted from the point of the last status saved instead of from the beginning.

COBOL–68 and COBOL–74

COBOL on DECsystem-10s is available in two versions: COBOL–68 and COBOL–74.

Table 8–1
COBOL–74 Support Levels

ANS–74 Module	Level Supported by COBOL–74	FIBS PUB21–1 Requirements for "High-level"
Nucleus	2	2
Table Handling	2	2
Sequential I/O	2	2
Relative I/O	2	2
Indexed I/O	2	2
Segmentation	2	2
Library	2	2
Debug	2**2	2
Interprogram communication	2	2
SORT/MERGE	2	2
Communication	**3	2
Report Writer	**4	2

**2 Debug module is functionally replaced by COBDDT
**3 Not supported
**4 ANSI–66 report writer syntax plus SUPPRESS statement

COBOL–68 is an implementation of the COBOL language based on the ANSI COBOL X3.23–1968 standard. COBOL–74 is an implementation of the COBOL language based on the ANSI COBOL XX3.23–1974 standard. COBOL–74 meets the high-level requirements of FIPS PUB 21–1 as tested by the FCCTS (Federal COBOL Compiler Testing Service), with the exceptions noted in Table 8–1.

BASIC–10

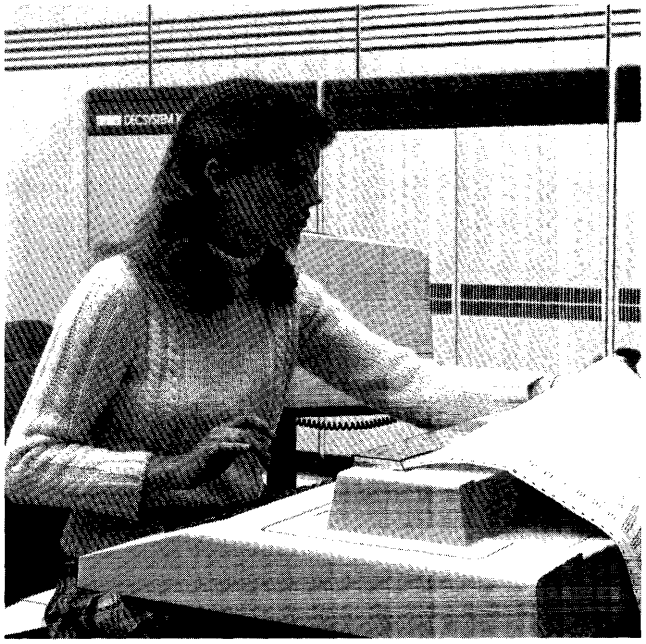
BASIC (Beginner's All-purpose Symbolic Instruction Code) is a problem-solving language that is easy to learn because of its conversational nature. It is particulary suited to a timesharing environment because of the ease of interaction between the user and the computer. This language can be used to solve problems with varying degrees of complexity, and thus has a wide application in the educational, business, and scientific markets.

BASIC is one the simplest programming compilier languages available because a small number of clearly understandable and readily learned statements are required for solving almost any problem. The BASIC language can be divided into two sections: one section of elementary statements that the user must know in order to write simple programs, and a second section of advanced techniques for more powerful programs.

The BASIC user types computational procedures as a series of numbered statements that are composed of common English terms and standard mathematical notation. After the statements are entered, a run-type command initiates the execution of the program and returns the results.

Extended features of BASIC–10 include:

- The PRINT USING statement — includes the leading asterisk, floating dollar sign, and imbedded comma
- Sequential-access file handling — for both data and text
- Random-access capability — for numeric and string files
- Simultaneous opening of up to nine files
- String handling ability — a full package which includes concatenation and other string functions
- Subroutine CHAIN features to other BASIC programs — includes restart capability at a specified program line
- I/O from/to any supported device, such as card reader, lineprinter, magnetic tape
- CRT support at multiple line speeds



ALGOL–10

ALGOL–10 is a scientific language designed for describing computation processing, or algorithms. It is a problem-solving language in which the problem is expressed as complete and precise statements of a procedure.

The TOPS–10 interactive editors and an integrated ALGOL debugger make it easy to code, test, and debug programs. Program modularity is available through block structure, subprograms, and separately compiled procedures.

ALGOL–10 is an implementation of the ALGOL–60 language. The compiler consists of a reentrant (sharable) segment and a data segment which varies in size depending on the size of the program to be compiled. The one-pass, single-phase compiler produces diagnostics and generates optimized object code.

ALGOL–10 language features include:

- Long real scalars, arrays, and procedures giving 62-bit mantissa using the double precision hardware
- String scalars, arrays, procedures, and byte manipulation allowing users to generate, manipulate, and input or output strings or individual bytes ranging in size from one to 36 bits
- Assignments within expressions
- Remainder operator
- Unique implementation of dynamic arrays
- Octal Boolean constants and integer/Boolean and Boolean/integer conversion functions
- WHILE statement and a convenient abbreviated form of the FOR statement

- Ability to call FORTRAN language subroutines and functions
- Identifiers up to 64 characters long

Block Structure
The ALGOL program structure is somewhat more complicated than high-level languages such as FORTRAN or BASIC, but offers compensating benefits. An ALGOL program has a number of hierarchically arranged blocks. A block consists of declarations and statements enclosed by the words BEGIN and END. Scope rules and declarations determine what variables are global or local to a block.

Block structure offers many advantages including easier implementation of top-down application design, code modularity, easy-to-read and understand programs, and increased protection from side effects.

Procedures
Algol procedures are subprograms that are useful in writing highly structured and easy-to-read programs. Code segments that are used more than once, and/or that can be viewed as a module of the larger program, can be separated into procedures. To aid top-down program development, procedures can be written, debugged, and compiled separately from the main program. Large structured programs frequently consist of a series of procedure calls.

Parameters can be passed to a procedure by value or by name. When an expression in a procedure call (an actual parameter or argument) is passed by value, a copy of its value is made available as a local, formal parameter (dummy variable) within the procedure. This is an efficient way to pass many expressions and to protect the calling block from side effects.

When an argument is passed to a procedure by name, any changes made to the value of the formal parameter in the procedure also changes the actual parameter as if it instead of the dummy appeared in the procedure body. Passing an array or a string argument by name instead of value saves memory space and allows the procedure to treat the argument as a global variable.

ALGOL supports recursive procedures; procedures can call themselves directly or indirectly to a depth limited only by the user's available memory space.

Compiler and System Features
The ALGOL compiler reports all source program errors on the user's terminal or on a listing device. The ALGOL–10 compiler adds these extensions to ALGOL–60:

- Long real type equivalent to FORTRAN's double-precision for more accurate real computations.
- External procedure can be compiled independently of main program.
- Convenience in loop implementation with WHILE and abbreviated FOR statements.
- Programmer can manipulate strings of various size bytes and can individually manipulate the bytes within a string via byte subscripting.
- Integer remainder function.
- Delimiter words can be represented in either reserved word format or as nonreserved words.
- Constants of type real can be expressed as an integer part or as a decimal part only.

OWN Variables
Special integer, real, long real, Boolean, and string variables called OWN variables have these properties:

- They follow the normal scope rules within a block.
- When control passes outside the block, the values are retained and still available when the block is reentered.
- Their values are initialized to zero, false, or null.

Switches
Switches provide the function of the CASE statement: program control jumps to various locations depending on the value of an expression. Switches automatically detect when the evaluated expression is out of range.

String Constants
String constants allow the user to refer to a string of ASCII characters within a program by using a variable name. The length of the string is limited only by available memory. String constants are typically used to convey messages to the program users or to assign value to string variables.

Object-Time System
The ALGOL–10 object-time system (ALGOTS) provides a basic input/output system so that the user can communicate with directoried and nondirectoried devices in ASCII and binary nodes. At run time, ALGOTS provides I/O processing, storage management, and debugging facilities.The object-time system includes a library of routines which can be incorporated in a user's program. Some of the routines are:

- A set of mathematical functions, including both single- and double- precision functions
- Maxima and minima functions
- String manipulation routines
- Bit field manipulation routines
- FORTRAN subprogram interface routines

The run-time facilities include:

- I/O with directory and nondirectory devices in both ASCII and binary modes — up to sixteen internal logic channels plus default terminal I/O channels are available.
- Storage management of the heap and stack — enables the program to borrow a temporary buffer for I/O, OWN arrays, and dynamically created byte strings. It also provides memory expansion when needed.
- ALGOL dynamic debugger — allows interruption of program execution, setting and clearing of breakpoints, examination and alteration of ALGOL variables that are in scope, examination of various system parameters, automatic typing of ALGOL variables after a breakpoint, examination of the code generated, and continued program execution from both a halt and an appropriate label.

APL

Two levels of this popular language are available. Both levels are full APL implementations with significant extensions. The basic version suits users who do not require the file I/O or the advanced APL functions. The extended version of APL, APL–SF, has all of the features of the basic version plus advanced features that substantially increase the range of applications for which it can be used. Both levels of APL feature:

- Full implementation with significant extensions
- Fast execution
- System functions/variables
- Four-way error trapping
- Error analysis and recovery
- Support of highly interactive applications
- Double precision arithmetic
- Integrated debugging features
- Workspace interchange features

APL–SF permits the user to obtain canonical string representations, create local functions, erase and classify names from a workspace, and perform various file I/O operations including ENQ/DEQ. APL–SF features:

- System variables with which the programmer can set tolerances and index origins and store accounting information
- File I/O that makes it easy to structure program data and interchange data files between other DECSYSTEM–20 languages such as FORTRAN and COBOL
- Input and output system commands with which the programmer can redirect terminal input and output to any file
- An operator that can solve linear equations, take the inverse of a matrix, or solve an overdetermined set of linear equations using a least-squares fit
- An operator to convert numeric data to a character string and enable the programmer to write user-defined functions to perform special output formatting and function editing
- An operator that makes it efficient to find the indices in a vectorfor which a particular boolean expression is true
- An operator that permits a character to be executed as an APL statement
- A convenient and efficient mechanism for formatting output data; for example, a entire table with associated text can be formatted in a single operation or a large matrix can be formatted with alphanumerics

APL uses one of the most concise, consistent, and powerful character sets ever devised. APL is especially suited for handling array-structured alphanumeric data. It is also used as a general data processing language and a mathematical tool.

APL allows programmer-defined functions and primitive language functions to be expressed with the same syntax. Thus, programmers can expand the capabilities of the language to handle the requirements of any application.

Data Structures

APL supports a variety of numeric and character data structures. They are:

- Scalars — a single numeric or character value with no dimensions
- Vectors — a one-dimensional array or character string consisting of any number of values
- Matrices — a two-dimensional array consisting of rows and columns
- Arrays with three to 16 dimensions

Interacting with APL

Programmers interact with APL using a hardcopy or video terminal. DIGITAL's LA37 hardcopy terminal includes an APL/ASCII dual character set. The LA37's keyboard is designed specifically for use with APL. For ASCII and console terminals, special APL characters can be represented by keyboard mnemonics.

System Commands and I–Beam Functions

Programmers can change system parameters, determine hardware or operational characteristics, and modify workspace parameters through system commands and I-beam functions. System commands control the operational environment in which an APL

session is conducted by allowing programmers to examine or change the state of the system. I-beams are APL functions used to communicate with the APL system to change user workspace characteristics and to report statistics about the workspace and the APL system.

Statements

A program consists of one or more lines called statements. There are two types of APL statements: assignment statements and branch statements. Assignment statements include calculation and input/output operations. Branch statements are used to restart a function or to handle the transfer of control from one part of a program to another. Branch statements are relevant only to programmer-defined functions.

An APL statement can contain:

- Identifiers
- Constants
- APL primitive functions
- User-defined functions

APL Statement Execution

APL language statements operate in either of two modes:

- Immediate or execution mode — in this desk-calculator mode, APL statements and expressions entered by the user are executed immediately.
- Function-definition mode — in this mode, APL programs and functions are developed, edited, named, and saved for later use.

Programmers can shift from one mode to the other. The syntax of the APL language is identical in both modes.

Debugging Tools

Function execution is suspended if an error occurs or if a stop vector is set. When execution is suspended, the name of the suspended function and the line number of the statement that would have been executed next are displayed. APL then awaits input in immediate mode. Programmers can perform any other APL operations at this time. The programmer can resume execution after fixing the problem and can observe function nesting. Programmers can also obtain an automatic display of the intermediate results of function execution. As a program tracing aid, the values computed by one or more function statements can be output each time those statements are executed.

APL allows programmers to suspend execution of a function from within the function itself. A stop control vector with a syntax similar to that of the trace vector suspends function execution just before execution of one or more specified statements.

Workspaces

An APL workspace is a buffer in the programmer's memory area that stores the functions, variables, values, and temporary results obtained while executing APL statements. Using APL system commands, workspaces can be saved, retrieved, and erased. They can be stored on a variety of devices, including disk, magnetic tape, and flexible diskettes.

A workspace can be saved in either memory-image or ASCII format. Workspaces saved in ASCII form can be created and edited with any DIGITAL-supplied editor.

File Organization

The APL file system allows the APL programmer to access data and program files on a variety of system devices. The file system is implemented as an integral part of the APL language and provides an interface to the TOPS–10 operating system.

The APL file system support is provided by:

- System commands for assigning, creating, closing, reading, writing, and renaming files
- File operators for byte pointer, input, and output functions

APL supports ASCII sequential and random access files. ASCII sequential data files can be read and written sequentially by any TOPS–10 language processor (e.g., BASIC, FORTRAN, MACRO). The system treats random-access files as random access memory. Programmers can directly access any byte in the file by specifying the individual byte or value to be read or written.

Error Analysis and Recovery

Instead of stopping execution, APL–SF error analysis and recovery permits the program to take remedial action. Error trapping features permit computer-assisted-instruction applications. A programmer can, for example, write a program that lets a student write a problem solution. The controlling program can intercept a student error.

Conversion Package

A complete package is available to convert IBM APL/SV workspaces and data files into APL–SF workspaces and data files. The package is written mostly in APL and is easy to adapt for use with other APL systems. The package is written according to the workspace interchange standard.

BLISS–36

BLISS–36 is DIGITAL's implementation language for

software development. BLISS is an optimizing, high-level systems-implementation language for the DECSYSTEM–20. It is specifically designed for building compilers, real-time processors, utilities, and operating system software. BLISS encourages the writing of highly structured programs that are easy to maintain.

BLISS has the features of a modern, structured high-level language combined with the flexibility of assembly language. BLISS can help systems programmers be more productive, shorten project development time, and lower maintenance cost.

The key features of BLISS are:

- State-of-the-art optimization technique to generate highly-optimized programs
- A full set of structured programming constructs including IF–THEN–ELSE, CASE, DO–WHILE, SELECT, and DECR statements
- Sophisticated macroprocessing capabilities
- Access to machine-dependent features including PSECTS, hardware registers, and machine-instructions (including UUO and JSYS)
- A linkage declaration that supports user-selected register conventions. Users can rebuild the object time system to support nonstandard register conventions.
- Precompiled source libraries similar to UNV MONINT, MONSYM, and UUOSYM

Compiling
The compiler can display a listing of errors and warning flags on the programmer's terminal, or can generate a listing with the errors and flags embedded. Listing can be printed as needed, and most syntactic errors are labelled as such.

Debugging
An interactive symbolic debugger supporting BLISS-style expression evaluation is used.

File Organization
Any file organization can be accessed using BLISS.

Compatibility with Other Languages
BLISS is not intended to replace the other high level languages such as COBOL, FORTRAN, or BASIC. Instead, it complements them. Programs written in other languages can call BLISS routines through the standard operating system calling sequence. BLISS programs can call routines written in a variety of other languages and use several linkage conventions.

CPL
CPL (conversational programming language) is a PL/I subset interpreter. It is a well-documented and easy to learn subset of ANSI–1976 PL/I.

At the user's option, statements are executed immediately or saved for deferred execution. A beginning programmer can start by executing simple computational statements and proceed to building programs.

Since CPL is an interpreter, a user can track his program very closely. Debugging features are included, such as source level breakpoints and program modification.

CPL includes the following features:

- ANS PL/I statements ALLOCATE, ASSIGNMENT, BEGIN, CALL, CLOSE, DECLARE, DEFAULT, DELETE, DO, END, FORMAT, FREE, GET, GOTO, IF, NULL, ON, OPEN, PROCEDURE, PUT, READ, RETURN, REVERT, SIGNAL, STOP, and WRITE
- Data types FIXED, FLOAT, CHARACTER, CHARACTER VARYING, BIT, BIT VARYING, POINTER, and arrays of these types
- Storage classes AUTOMATIC, STATIC, CONTROLLED, and BASED
- Recursive procedure support
- Almost all ANS PL/I arithmetic, mathematical, string-handling, array, and storage control built-in functions
- STRING, SUBSTR and UNSPEC pseudovariables

Immediate Mode
The CPL immediate or desk calculator mode is integrated into the system so that the user can easily move between programming and calculating.

Almost any CPL PL/I language statement can be typed without a line number for immediate mode or with a line number for deferred execution as part of a program. A user can do simple computations to get immediate answers, and can even assign intermediate results to variables and use the variables in further computations. Immediate mode is thus a useful for debugging.

Program Creation
The user does not need any editor or utility to create a CPL program. Program statements are typed to CPL; insertions, changes, or deletions are done easily.

CPL provides automatic syntax checking of statements. When a statement is typed, CPL immediately checks for syntax errors. If there is an error, CPL tells where the error occurred. The user then has the opportunity to correct the statement and go on. This means that a user gets immediate feedback on syntax errors rather than waiting for a compilation to provide a list of errors. Of course, a user need not use the CPL

program editing facility. Any editor or file manipulation facilities can be used to create or modify a program. The file can then be loaded into CPL and any erroneous statement will be listed with an error message that indicates the problem.

Debugging

CPL is the only PL/I language processor available that provides true source-level program debugging. A programmer can use these steps to debug a program:

1. The programmer loads the program, sets breakpoints on any statement, and executes the program.
2. The program stops on an error, a breakpoint, or at programmer command.
3. The programmer uses CPL statements in immediate mode to isolate problems. Variables can be examined or changed. I/O can be done. New variables can be created to store intermediate results.
4. The programmer makes source modifications to the program. Statements can be added, deleted, or modified, and breakpoints can be set or cleared.
5. The programmer continues execution, starting after the current halt in execution.

9
Data Management and Application Products

In addition to the standard file management facilities included with TOPS–10, optional data management and application products are available.

DBMS is a CODASYL compliant data base management system that lets users organize and maintain data in customized data bases and provides rapid and convenient access to the data. IQL is an interactive query language for information retrieval and report writing. IQL can access DBMS files. SORT/MERGE is a -10 sort utility that operates stand-alone or with COBOL or FORTRAN programs to reorder the records of files into new sequences or to merge sorted files into a single sorted file.

Two application products, COGO–10 and PCS–10, are discussed. COGO–10 includes a geometric language for solving problems in plane coordinate geometry. PCS–10 is a project control system that analyzes critical patch or procedure networks and generates a number of resource, cost, and critical path reports.

COMPONENTS

The data management products detailed in this section are:

- DBMS — a CODASYL data base management system
- IQL — a interactive query language for information retrieval and report writing
- SORT/MERGE — a sorting and merging utility for TOPS–10 files

The optional application products discussed below are:

- COGO–10 — a tool for solving problems in plane coordinate geometry using a geometric language.
- PCS–10 — a project control system that analyzes critical path or procedure networks

The comprehensive data management system that is part of TOPS–10 and included with all DECsystem-10s is discussed in Section 4 of this technical summary under the headline The File System. Access to data files from various programming languages is described in Section 8.

DBMS

DBMS is an optional TOPS–10 CODASYL compliant data base management system that lets users organize and maintain data in customized data bases and provides rapid and convenient access to the data.

DBMS integrates related processes and data structure and is used when traditional file management techniques would be difficult, costly, inadequate, and/or error prone. DBMS features:

- COBOL and FORTRAN interfaces
- English-language interface for inquiry and report generation through the IQL package (see below)
- Easy SCHEMA/SUBSCHEMA definition and update
- Full journaling and data base recovery capability
- Protection against unauthorized data base access with centralized control of privacy
- High throughput

CODASYL Compliance

DBMS provides software to define, access, and maintain data in the network structures of an integrated data base. DBMS is based on the CODASYL Data Base Task Group Report of April 1971.

Data Description Process

Data structures can be established in either a hierarchical form or in networks with multilevel relationships. Relationships can exist within files or between one or more files (areas) with no fixed limit on the length of chains (the TOPS–10 file system is used to construct data base areas). The number of relationships in which any record can participate and the size of the record is normally limited only by the design constraints of the application.

Data Manipulation Process

Data base records can be referenced through a set of data manipulation statements included in COBOL or FORTRAN application programs. These statements include the ability to store, modify, and delete fields and records that are of specific interest to an authorized application user. Also included is the ability to insert and remove records within structural relationships. It is possible (by the data definition process) to directly access specific records by symbolic keys or by movement through structural relationships.

DBMS Modules

TOPS–10 DBMS includes these elements:

- DDL (Data Description Language), including DMCL and subschema DDL — used by a data base administrator to create and maintain procedure-language-level descriptions, records, areas (data files), and sets (interrecord relationships). The data base descriptions (schema/subschema) are established and maintained by the DDL compiler, SCHEMA.
- DML (Data Manipulation Language) — used by an application programmer writing in COBOL or FORTRAN to store, retrieve, modify, or delete data and to insert and remove interrecord data relationships. The DML comprises an extended verb set which is added to the COBOL host language and to a preprocessor for FORTRAN programs. COBOL programs which include Data Manipulation Language statements are processed directly by the COBOL compiler.
- DBCS (Data Base Control System) — a run-time interface between a COBOL or FORTRAN program and the TOPS–10 operating system. The DBCS exists in a specific module within the COBOL object-time system (LIBOL) and the FORTRAN object-time system (FOROTS). Recovery of failed DML updates is via the journaling facility.

DBMS Utilities

The DBMS utilities help the data base manager run and maintain the data base. The utilities include:

- DBMEND — a recovery utility that provides rollback and rollforward of selected data base transaction.
- DBINFO — a utility that selectively dumps information from a data base. It can also be used to generate statistics about the data base, a DDL cross

reference listing, and other descriptive information about the DDL program.

- STATS — a component with DBCS used to obtain run-time information about each DML verb and statistics about data base page reads and writes.

IQL

IQL is an optional TOPS–10 interactive query language for information retrieval and report writing. It takes query requests written in English, such as formats, reads one or more input files, and process the data according to the request. IQL queries are groups of easily written report generation commands. IQL can access TOPS–10 files and interface to DBMS for a powerful, fully-integrated data base management system. IQL can interface to the operating system's file management system and to DBMS. With them IQL is a fully integrated data base management system that can quickly satisfy data retrieval and report generation requirements.

IQL features:

- Multiple input files (DBMS, ISAM, SEQUENTIAL).
- Extensive record selection.
- Sorting.
- Conditional processing and/or conditions can be strung together; parentheses can be nested nine deep.
- Built-in summary statements.
- Complete report formatting capabilities, including multiple across labels and special forms.
- Multiple reports up to nine, standard, and expandable.
- Files output in original or new format.
- Matrix reporting by manipulation of summaries or individual items.
- Powerful computation capabilities.
- Built-in summary statements for tallies, totals, and averages.
- Update capability.
- Dictionary pre-sorting of file, record, and item information, including printing column titles and pictures.
- Default automatic formatting of reports, including field alignment, dates, paging, and column.
- Interactive or batch modes of operation.
- Exits to user-written modules.

Traditionally, data manipulation and reporting has been left to programs written in COBOL, FORTRAN, or assembly language. This method incurs high programming costs and delays in obtaining report information. IQL, however, is an inquiry system that can quickly extract, summarize, reorganize, report and copy file information. The accompanying table illustrates the use of IQL ad-hoc inquiry and reporting.

Interactive Mode

In interactive mode, IQL operates under control of terminal front-end modules that permit the terminal user to:

Table 9–1
IQL Ad Hoc Reporting

Customer Name	Y-T-D Purchases	Sales	Difference
Manfredini Violin Co.	$1,233.67	$ 1,357.03	$123.36
Globalex	$2,500.00	$ 2,750.00	$250.00
Lakeside Homes Inc.	$1,211.00	$ 1,332.10	$121.10
Farfield Motors	$9,000.00	$ 9,900.00	$900.00
Hubert Oil Company	$ 600.00	$ 660.00	$ 60.00
Bee Drill Service	$1,000.00	$ 1,100.00	$100.00
Investment Inc.	$ 364.75	$ 401.22	$ 36.47
Energy Resources Inc.	$8,000.00	$ 8,800.00	$800.00
Geo Bank Fisheries	$9,164.00	$10,080.40	$916.40
Merrimack Valley Sales	$6,000.00	$ 6,600.00	$600.00
Apple Orchard Inc.	$ 671.00	$ 738.10	$ 67.10
Craig Publishing	$5,115.50	$ 5,627.05	$511.55
Final Summaries			
Y–T–D Total: $44,859.92			
Sales Total: $49,345.90			
Difference Total: $ 4,485.88			
End query execution			

- Write, store, retrieve, or change queries
- Define and interrogate dictionaries
- Define dictionaries reflecting SCHEMA files for DBMS data bases
- Browse or update sequential or indexed sequential input files
- Create sequential output files
- Accept raw data input
- Operate other IQL system modules
- Display snapshot reports on the terminal

Deferred Mode
In deferred mode the IQL system provides powerful retrieval selection, report formating, sorting, computation, summarization, and data file writing capabilities. Up to three input files can be read. The input files can be DBMS data bases, sequential or index sequential. They can contain fixed or variable length records, and one or more record types. Input data files are queried in their original format and can be in SIXBIT or ASCII mode. All standard data item types are permitted. Only sequential or index sequential input files can be updated; DBMS databases can not. Sequential output files can be generated that either "mirror" the format of the primary input data file or assume a new format as specified by the query.

Queries can be stored in text files or in executable form for later reuse.

IQL Statements
An IQL selection statement can combine many conditional tests connected by AND and OR logical operators and clarified with parentheses. An IQL computational statement can include addition, subtraction, multiplication, and division with parentheses. Built-in summary statements for tally, total, average, maximum, and minimum calculation can be controlled by data item breaks. A random number variable is built in.

DBMS Files
IQL accesses data files under control of a data dictionary that describes the format of the file and the location, default display pictures, and column titles of each item in the data record(s). For DBMS data bases the dictionary also includes pertinent information about record names, set name, and area names. Password protection can be applied to individual data items or groups of items.

Use of DBMS data base files requires that the SCHEMA file be present as well as the data base files. Additional security for DBMS data bases can be provided with privacy locks. IQl's queries operate in stages delimited by SORT statements. IQL can sort mixed ascending or descending on data items or calculated fields.

Report Formating
IQL report formatting can be automatic, or the user can specify custom report formats. Data item placement in a report is specified by the user. Both multiple print lines per input data record and multiple input data records per print line are permitted. Special-form reports such as mailing labels and checks can be produced easily. Up to 99 multiple reports can be generated at one time.

SORT/MERGE
SORT/MERGE is an optional TOPS–10 sort utility that operates stand-alone or with COBOL or FORTRAN programs.

SORT/MERGE reorders the records of: EBCDIC files, ASCII files, SIXBIT files, and Binary files produced by COBOL or FORTRAN in a sequence determined by the sorting parameters specified by the user.

SORT/MERGE automatically controls the use and allocation of disk workspace and memory work space. The user can also specify memory limits. SORT/MERGE provides error diagnostics and statistics upon completion.

The MERGE capability permits the merging of sorted files into a single sorted file. This function can be used either stand-alone or with COBOL.

COGO–10
COGO–10 is an optional TOPS–10 tool, including a geometric language, for solving problems in plane coordinate geometry. It is used in such fields as land surveying, highway design, right-of-way surveys, bridge geometry, and subdivision work.

The COGO language is the part of the tool that the user works with directly. Most of the COGO–10 command names and command codes are identical with those of SELLS COGO.

The language consists of common engineering terms, and no computer experience is required to use it. Problem definitions can be stored in a file of commands and executed at a later time, or can be entered using a terminal keyboard interactively. The commands are identical either way.

COGO has commands for:

- Starting and ending a COGO job or changing the I/O device
- Maintaining tables (lists of related points such as road alignments or property lines)
- Intersecting existing lines or figures to calculate a new point

- Calculating and/or storing one or more points
- "Locating" in traverse work
- Alignment and spiral functions
- Maintaining compatibility with the previous COGO so that existing input can be used
- Outputting information generated by COGO

PCS–10

PCS–10 is an optional TOPS–10 project control system that analyzes critical patch or procedure networks and generates a number of resource, cost, and critical path reports.

PCS–10 can process data for any critical path network developed an an IJ, CPM, or precedence technique. The fixed format input can be entered on a variety of media. The system includes a terminal-oriented editor for input from a CRT or hard-copy terminal.

PCS–10 has the following features:

- The time required to perform a work item can be expressed in seven different time units.
- Seven different days can be designated as the starting day of the work week associated with each work item.
- The work week of each work item can be one to seven days.
- Any one of three distinct calendars can be used for each work item.
- The actual dates can be specified for each work item.
- One of five types of "scheduled" dates can be specified for the start and completion of each work item.
- Two "early" and two "late" dates can be computed for each work item.
- Two types of duration can be specified for each work item.
- Two types of percent complete can be specified for each work item.
- Two kinds of "float" can be computed for each work item.
- Three different kinds of codes can be specified to be associated with each work item.
- Three kinds of costs can be specified for each work item.
- An IJ/CPM or a precedence network can be accepted for processing.
- An IJ/CPM network is internally converted into a precedence network for processing.
- Three different relationships can be specified between any two work itmes when using precedence input.
- Loops are detected and identified.
- A time delay can be specified that becomes an intrinsic part of the relationship between two work items.
- All network calculations are based primarily on three calculation dates.
- 240 "milestones" can be specified.
- Fourteen different types of output reports can be produced on request, including four different kind of system runs that can be specified, and a network that can extend over a period of 2911 calendar days, slightly less than eight years.

10 Communications

TOPS–10 communications products are a unique combination of hardware and software providing flexible networking capabilities to numerous user communities. These products distribute computing power over various network topologies and can use several network protocols.

TOPS–10 communications software (TOPS–10 Networks), remote concentrators (DN200s), IBM-protocol emulator/terminators (TOPS–10 2780/3780 ETs), and TOPS–10-distributed processing software (DECnet-10) comprise TOPS–10 communications products. These products let TOPS–10 hosts communicate with other TOPS–10 hosts, other non–TOPS–10 DIGITAL computer systems, TOPS–10 remote stations, and IBM computer systems, both host mainframe and IBM-like remote job entry stations.

Network Concepts

Independent of their communications link, users in a TOPS–10 network can access a central site or remote site with ease and efficiency. Remote users employ the same commands and facilities as users at the host sites and can communicate readily with multiple hosts and other remote stations.

Where the network contains more than one TOPS–10 host, the user can choose which host to access with a SET HOST command. The user can also choose a remote node for output with the LOCATE command, and view the status of network nodes with the NETWORK command. The NETWORK/TOPOLOGY command lets the user view the topology of the network. The WHERE command finds the network node where a specified device exists. The user can communicate with other network users or transfer files across the network with system utilities such as PIP.

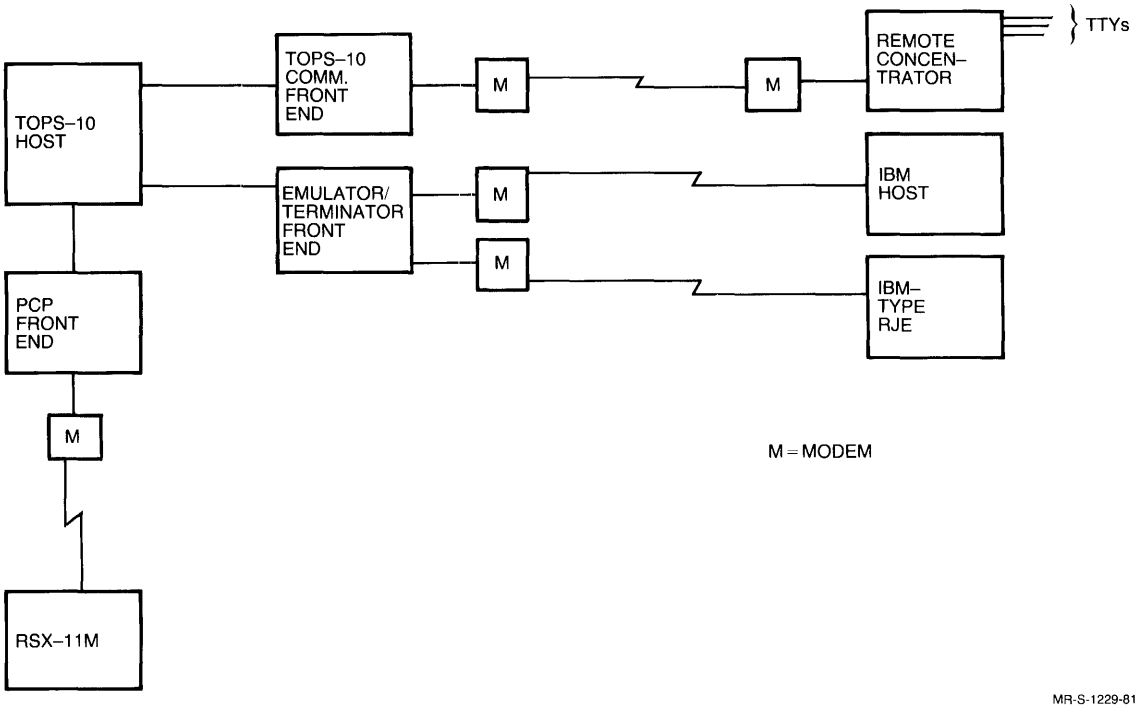


Figure 10–1
TOPS–10 Communication Products

```
.net RET

Node KL1026 (26) RZ124A KL 1026/1042 10-29-80
Node COMET (70) DN200 V22(146) 10-Sept-80
Node ENCORE (32) DN87S V22(146) 10-Sept-80
Node CTCH22 (22) DN82 V22(146) 10-Sept-80
Node K12102 (20)
Node JINX (134) DN20 V22(146) 20-June-80
Node NOVA (31) DN87S V22(146) 10-Sept-80

.net/topo RET

Node COMET (70) 32(10)
Node ENCORE (32) 26(10) 70(10) 31(10)
Node CTCH22 (22) 134(10)
Node JINX (134) 26(10) 22(10) 31(10)
Node NOVA (31) 26(10) 134(10)
```

Figure 10–2
Sample NETWORK/TOPOLOGY Command

A user at a remote site can enter programs, text, or data at a remote terminal, store the data at the central site, and later print it at the local station. The remotely located user can further communicate with other non-central hosts, and with other users at other remote sites. This facilitates user communication with the central location and enhances the capabilities of the remote site. These capabilities are made possible by the use of standardized data transmission techniques, asynchronous and synchronous communications, and specialized protocols.

Data Transmission Techniques

The transmission of coded information between terminals and computers, or between computers, is the key capability inherent in all communications and computer network systems. There are several transmission techniques used to move data from place to place. The technique used by the DECsystem-10 is serial data transmission. Serial data transmission has two basic forms: synchronous and asynchronous.

Serial Data Transmission

Serial data transmission moves data in a serial stream of bits, one bit following the other. Serial transmission can move data character-by-character or in blocks. Serial data transmission can be either synchronous or asynchronous. With synchronous communications, characters are framed with a pair of mutually synchronized clocks; one in the transmitter and the other in the receiver. With asynchronous communications, character framing is alone with start and stop bits.

Asynchronous Transmission

Asynchronous transmission is used for connecting interactive terminals to the DECsystem-10 host and its remote stations. Asynchronous communications provides for a low cost connection when the line usage is intermittent. This is typical of terminals that are being used by interactively.

The DECsystem-10 supports a variety of ASCII terminals, both hard copy and display. Typical supported terminals include the LA–38, DECwriter-III and VT–100 display terminals. A variety of other ASCII terminals can also be used.

Terminals can be connected to line interfaces attached to the DECsystem-10 using either current-loop or EIA connections. For terminals that are remote from the host, two options are available. A remote terminal concentrator can be provided for services several nearby terminals, or dial-up lines with modems can be used. Dial-up lines can also be connected to a remote station.

TOPS–10 supports full duplex operation of asynchronous lines. This means that the operator of a terminal on a TOPS–10 system can type on his terminal at the same time that the system is sending data. Thus, skilled operator need not wait for prompts when typing large amounts of data or a long sequence of commands.

Synchronous Communications

Synchronous communications is used by the networking software to provide high speed data transmission over local, leased or switched lines. Synchronous communications is always done with some type of clocking modem, which improves transmission efficiency by eliminating the need for start and stop bits. In TOPS–10 networks, synchronous lines are used for connecting hosts and remote stations together.

TOPS–10 Network Protocols

TOPS–10 implements a set of protocols that are known collectively as the TOPS–10 Network Protocols, or ANF–10. These protocols are layered to provide various network services.

TOPS–10 uses the DDCMP (DIGITAL Data Communications Message Protocol) protocol for data transmission over synchronous lines. DDCMP provides for error detection, correction and retransmission to provide error-free communications over imperfect communications lines. Protocols, such as DDCMP, that are used over a single physical link are called data link protocols.

TOPS–10 uses NCL (Network Command Language) for both device control and routing. NCL is a protocol that is layered on top of a data link protocol, such as DDCMP.

NCL device control provides support for line printers, card readers terminals and task-to-task. Terminal support allows terminals connected to a TOPS–10 host to "set host" to any TOPS–10 system connected to the network, giving the terminal the characteristics of a locally connected terminal. Line printer and card reader support allows a TOPS–10 host (or hosts) to use a remote line printer or card reader as though it were a local device. The task-to-task device allows two cooperating programs to communicate with each other.

NCL routing provides a startup protocol to allow nodes to be connected and disconnected from the network dynamically. It provides routing services to allow nonadjacent nodes to communicate with each other. NCL routing also provides error recovery code to resend messages that get lost during transmission between intermediate nodes.

COMMUNICATIONS PRODUCTS

TOPS–10 communications software (TOPS–10

Networks) and remote concentrators (DN200s) use Network Control Language (NCL) and DDCMP (DIGITAL Communications Message Protocol) to create the common network structure in which users communicate. The IBM-protocol emulator/terminator (TOPS–10 2780/3780 ET) uses IBM's binary synchronous protocol to communicate with IBM-type remote job entry stations and with remote IBM hosts. When communicating with an IBM 360 or 370 host, the 2780/3780 software is in emulation mode; when communicating with a remote job entry station, the 2780/3780 software is in termination mode.

With the DECnet-10 product, DIGITAL Network Architecture (DNA) provides the background on which the software is built. The architecture is completely modular and designed to handle a broad range of application requirements. With DNA, a network node can operate as a transfer station, as a front end, as a terminal concentrator, or as a host.

TOPS–10 Networks

Advanced Network Features (ANF–10) can be built upon TOPS–10 systems linked with remote terminal concentrators and remote job entry stations.

Independent of their communications link, users in a TOPS–10 network can access any DECsystem-10 or remote station with ease and efficiency. Remote users employ the same commands and facilities as users at host sites and can communicate readily with multiple hosts and other remote stations.

A network contains both local and remote sites and can link together data collections stations, remote control stations, remote concentrators, remote terminals, and remote job entry stations with line printers and card readers.

Communication software in the TOPS–10 system is an integral part of the system. The software enlarges and enhances the capabilities of the system, extending the system's reach and providing remote links to the most distant users. The TOPS–10 interface to the user is completely transparent. No matter where the user is on the network when connected to a TOPS–10 system, the system looks the same. The user can his entire effort into developing whatever application is required at the local site; the TOPS–10 communications products take care of all the communications needs and activities.

TOPS–10 synchronous communications can extend the capabilities of the DECsystem-10 through the use of remote stations. Remote stations and DECsystem-10s can be connected by simple and/or complex topologies, with up to a maximum of 63 nodes. These remote stations can include a line printer, a card reader, a console terminal, and, depending on the type of remote station, up to 32 asynchronous command terminals. Use of peripheral devices at various stations provides the user with increased capabilities. For example, data can be collected from the various remote stations, compiled and processed at the central site, and the results of the processing can be sent to all contributors of the data.

TOPS–10 synchronous communication provides error-correcting, high-speed paths among TOPS–10 systems and remote stations. The high-speed synchronous transmission is message-by-message, not character-by-character, as in lower speed, asynchronous transmission. Transmission errors are detected using cyclic redundancy checks (CRC–16). Data errors are corrected through retransmission of the damaged block. With TOPS–10 network software and supplementary TSKSER software, the homogeneous TOPS–10 network supports file transfer and network command terminals, as well as remote stations.

ANF–10 has many features for increased network availability and flexibility. These features include multipathing (the ability to have several paths between two points), dynamic reconfiguration (automatic redefinition of network topology and rewriting of messages if a node fails), and route through (the ability to send messages by means of intermediate nodes).

TOPS–10 2780/3780 ET

TOPS–10 provides communication with IBM systems and remote stations that use IBM's 2780 and 3780 protocols. The TOPS–10 2780/3780 software product is well suited to performing batch-mode bisynch operations. At a remote job entry station, the TOPS–10 2780/3780 software supports a card reader for input and a printer for output.

The TOPS–10 2780/3780 ET emulates and/or terminates Model 76 DATA 100 units with 2780/3780 features. Equivalent units can also be utilitized, but it is the customer's responsibility to prove equivalency. The software enables a DECsystem-10 equipped with a DN20 front end to:

- Send 2780/3780-type remote entry jobs to an IBM 360/370
- Process jobs submitted from 2780/3780-type terminals

In emulation mode, the DECsystem-10 connects to an IBM 360 or 370. The DECsystem-10 appears to the IBM system to be a DATA 100 emulating an IBM 2780 or 3780 remote job entry station. In emulation mode, the IBM host acts as the host. The IBM host must be running one of the allowed operating systems:

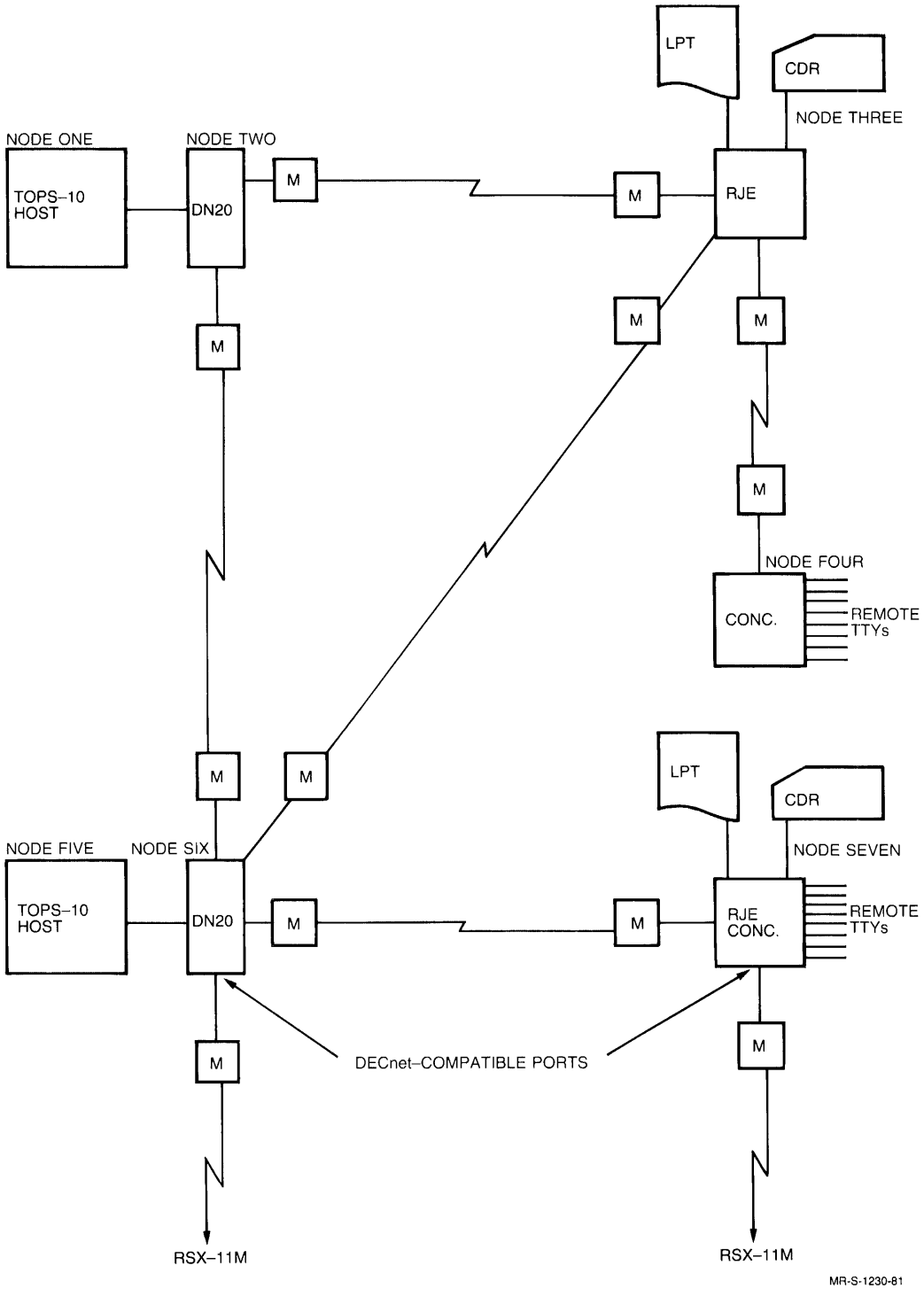


Figure 10–3
Complex Topology

- OS/VS2 (SVS) HASP II, Version 4.0
- OS/VS2 (MVS) JES2
- OS/MVT HASP II, Version 3.1
- OS/MVT ASP Version 3.1
- OS/VS2 (SVS), Version 3.2

A DECsystem-10 user can submit a disk file containing IBM JCL and ASCII data to the IBM system as a batch job. The software translates the data to EBCDIC before transmitting the job to the IBM system. Any output data returned to the DECsystem-10 is translated to ASCII and printed. In emulation mode, the software does not handle special forms.

In termination mode, the DECsystem-10 acts as the host. The TOPS–10 system connects to a Model 76 DATA 100 running either 2780 or 3780 software, or equivalent units. The 2780/3780 remote station user submits a DECsystem-10 batch control file on cards. The log file from the batch job is automatically returned to the remote station for printing. Other job output in printed form can also be routed back to the remote station printer.

DECnet-10

The DECnet-Compatible Port (DCP) is a software product that allows a suitably configured DECsystem-10 or DECsystem-10 network to communicate with PDP–11 systems running DECnet-11M.

The DCP offers task-to-task communications with RSX over synchronous communication lines. It operates by translating between subsets of the TOPS–10 Network Control Language (NCL) protocol and the Phase II DIGITAL Network Architecture (DNA) protocols. The DCP communicates only with DECnet-11M.

User programs access the DCP through the TOPS–10 network interface. TOPS–10 user programs written in languages that support device specifications (for example, FORTRAN, COBOL, and MACRO) can access the network. The DCP provides task-to-task communications facilities only. With the DCP, a TOPS–10 user program can exchange messages with a cooperating user program an adjacent DECnet-11M system.

The messages sent and received by the two user programs can be in any data format mutually acceptable to the two user programs. To TOPS–10 user programs, the DECnet-11M node appears to be a TOPS–10 Network node; to the remote DECnet-11 system user programs, the TOPS–10 system appears as part of the physical link.

The DCP supports the DIGITAL Data Communication Message Protocol (DDCMP) for full-duplex transmission in point-to-point operation using serial synchronous facilities. DDCMP provides error detection, error correction, and physical link management capabilities.

A DECsystem-10 typically supports a network of front ends and remote stations employing the TOPS–10 network native protocol. The DCP is an adjunct to that network, providing communication with one or more DECnet-11M systems.

11
Support Services

DIGITAL offers comprehensive support services to help customers before, during, and after system installation. DIGITAL's sales force is the primary contact for all products and services.

The support DIGITAL provides customers is apparent from the first. Our sales representatives work closely with customers. They study the application with the customer and determine specific computing needs. Software and hardware specialists are available to supplement the sales rep's product knowledge. These specialists, trained to design systems using DIGITAL's standard and special products, can be called in to answer specific questions.

Once the exact system requirements have been determined the sales rep helps the customer select a system configuration. Site requirements such as adequate floor space, electrical capacity, air conditioning, and humidity control are reviewed. Customers can choose among various Field Service and Software Service maintenance plans to suit individual needs and budgets.

If the application is complex, the customer and DIGITAL's Software Service organization can prepare a Customer Support Plan (CSP). The CSP can consist of Software Product Services, Educational Services courses, hardware maintenance requirements, and software consulting services. The CSP identifies the customer's needs; the purposes, benefits, and details of the services recommended; how the services will be delivered; and how much they will cost.

Even before the DECsystem-10 arrives, customers can train their personnel through DIGITAL's comprehensive educational programs. When a system is purchased customers obtain training credits that they can apply to the cost of courses.

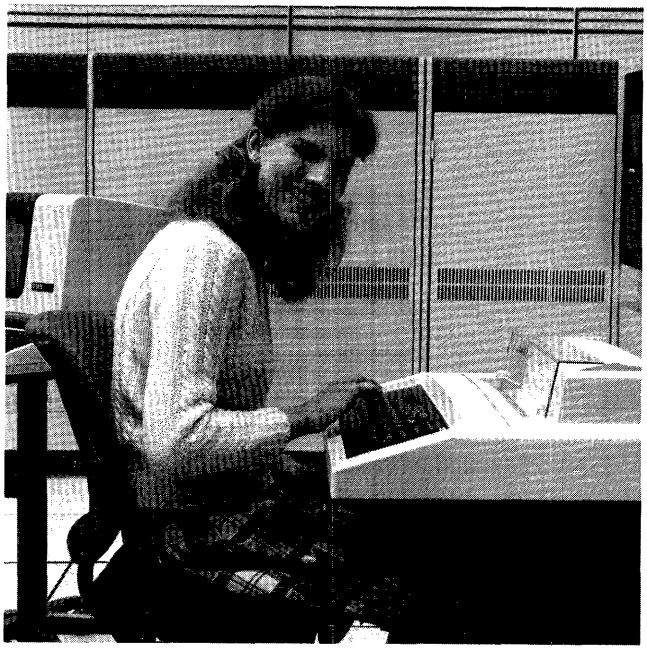
When a system is delivered, DIGITAL's hardware Field Service and Software Support organizations are on hand to ensure smooth installation. Specialists install hardware and software and run tests to determine that the system has been installed correctly and performs properly.

Following installation, DIGITAL's support organizations are available to help with special needs that may arise both during and after the warranty period.

Installation

Upon system delivery DIGITAL's Field Service account representative schedules installation of the hardware components. During installation Field Service engineers supervise the uncrating and placement of equipment, cable connection, and power-up of components. They test the hardware by running a diagnostic package and, once hardware reliability is confirmed, they coordinate with software support personnel to install and test the operating system.

Finally, DIGITAL Field Service and Software Services complete forms that certify successful installation, and the customer acknowledges system acceptance by signing the Field Service Labor Accounting and Reporting System form.



Software Services

DIGITAL's Software Services organization specialists are committed to maintaining a high level of support for TOPS–10 software. They have the knowledge and experience necessary to analyze the user's needs and to identify and deliver the DIGITAL services that will help satisfy those needs. In addition to local software specialists, backup support from regional and corporate levels is available when necessary. DIGITAL's total software resources and expertise are available to support the TOPS–10 and its various dependent products.

Software Warranty

TOPS–10 is a DIGITAL-supported software product. A DIGITAL-supported software product is engineered according to corporate quality standards, operates in accordance with a Software Product Description (SPD), and carries DIGITAL's commitment to provide

support services for the product. TOPS–10 is a DIGITAL-installed product that must be installed by a qualified DIGITAL representative to qualify for software support.

DIGITAL-supported software products receive a 90-day warranty period following installation. If, during the 90 days of warranty, a problem with the software is encountered that DIGITAL determines to be caused by a defect in the current unaltered release of the product, the following remedial services are provided:

- If the software is inoperable, DIGITAL will apply a temporary correction or make a reasonable attempt to develop an emergency by-pass.
- DIGITAL will help the customer prepare a Software Performance Report (SPR). With an SPR, users can report problems with, or suggest enhancements to, DIGITAL's software or documentation.

After the initial purchase of a DIGITAL-supported product license, additional copies may be purchased. These can include support services or can be purchased as a "License-to-Copy only," in which case neither media nor support services are included.

TOPS–10 includes standard services as defined in the TOPS–10 SPD. See your sales representative for these details.

Software Product Services

After the 90-day warranty period two levels of Software Product Services are available to provide continued software support and maintenance. Designed to complement DIGITAL hardware services, these Software Product Services offer the most comprehensive post-warranty support in the industry:

- Software Product Updates. This service is for customers who install their own software. It includes Software Updates on the user's choice of available

media and the most recent documentation. No support services are provided but are available upon request.

- Self-Maintenance Service for Software. This contractual service includes SPR forms, a subscription to software product and documentation updates, and a newsletter that provides up-to-date information on the software product.

Professional Services

Whenever expert software assistance is needed, DIGITAL's software consultants are available. These software specialists are experienced designers and writers of custom software who can tailor DIGITAL software to specific needs. Their expertise can be applied to any phase of an application from analysis through implementation.

Software specialist services are available on a resident or per-call basis:

- Resident service is for users who need full-time on-site support. Resident consultants are particularly useful in new complex installations or in critical, long-term projects. Residents are available for a minimum of six months; however, arrangements can be made to extend the length of service to suit individual needs.
- Per-call service is for customers with irregular or infrequent consulting needs. Per-call (hourly) services are ordered as needed and generally extend from a few days to a few weeks.

You can learn more about DIGITAL's software services by contacting your local DIGITAL sales office.

Educational Services

DIGITAL provides comprehensive educational programs to train users before, during, and after system installation. Instruction in system management, operations, hardware, and software is by trained specialists at DIGITAL's worldwide training centers. Special on-site training and custom courses can also be arranged.

Course Options

Courses fall into three general categories:

- Generic Computer Courses. These provide a technical foundation for personnel who have little computer experience.
- Software Systems Courses. These are designed to train users, programmers, and operators to efficiently and knowledgeably use DIGITAL's operating systems, languages, and utilities. Courses are available for both beginning and advanced users. The student is assumed to have general computer and programming knowledge.
- Hardware Courses. These are designed for customers who intend to service their own equipment or want a general understanding of the components in their system. Courses in general hardware familiarization, hardware troubleshooting, and hardware maintenance are offered.

A generic, software, or hardware course may be a lecture/lab series, taught at DIGITAL Training Centers on regularly scheduled bases, an on-site course available by arrangement, or a packaged course.

- On-site courses. Educational Services can conduct group training courses at any convenient location, such as a customer's office or a company's training center. On-site instruction eliminates travel expenses and allows DIGITAL instructors to emphasize points of particular value to individual applications and operations.
- Packaged Courses. Designed for students who wish to learn computer fundamentals and TOPS–10 at a self-paced rate, these packaged courses are portable, self-contained, and modular in format. They are available either as audio/visual courses or self-paced instruction (SPI) workbook courses.

Audiovisual (A/V) courses use a combination of filmstrip/tape or videotape and workbooks. Among the A/V courses offered are Introduction to Data Communications and Introduction to Digital Logic.

Self-paced instruction courses use a workbook with explanatory text, examples, and exercises. Among the SPI courses available are DECsystem-10 DBMS Concepts, DECsystem-10 Operator Training, and Programming in FORTRAN.

For users with special needs Educational Services can create a course tailored to unique applications, needs, and schedules. These can be completely new or modifications of existing courses. Custom courses can be conducted at a DIGITAL Training Center or at a user's training center. Contact your sales representative for full details.

TOPS–10 Courses

DIGITAL Educational Services offers a comprehensive software training program designed to address the needs of all levels of TOPS–10 users. The courses consist of both self-paced and lecture/lab courses that stress practical job-relevant skills required by TOPS–10 users, applications programmers, and system programmers.

There are 11 TOPS–10 courses. Users' job requirements dictate course content, and students can choose courses from the appropriate sequence. Figure 11–1 is a course flowchart. The option to combine lecture/lab and self-paced instruction courses gives students the freedom to learn at their own speeds, at their job sites, and to benefit from instructor aid and the practical experience obtained in classroom and laboratory sessions.

The following are brief descriptions of some of the DECsystem-10 courses shown in Figure 11–1.

TOPS–10 User provides a DECsystem-10 software and hardware overview and teaches students how to use the program development features of the system. Disk file organization, system utilities, command language, and the batch system are covered.

TOPS–10 Operator teaches students operator duties and the basic system procedures: startup, shutdown, recovery, backup, and file restoration.

TOPS–10 Assembly Language Programming covers the DECsystem-10 instruction set and the MACRO Assembler. Students learn to write programs that use monitor calls and the MACRO library and to use Dynamic Debugging Technique.

TOPS–10 COBOL teaches how to write COBOL applications programs under the TOPS–10 operating system.

TOPS–10 System Programming covers system controls, the accounting package, and advanced system calls.

TOPS–10 Applications Programming Techniques emphasizes the techniques that can be used to take advantage of system facilities with attention to the best overall design of an application in the timesharing or production environment provided by TOPS–10.

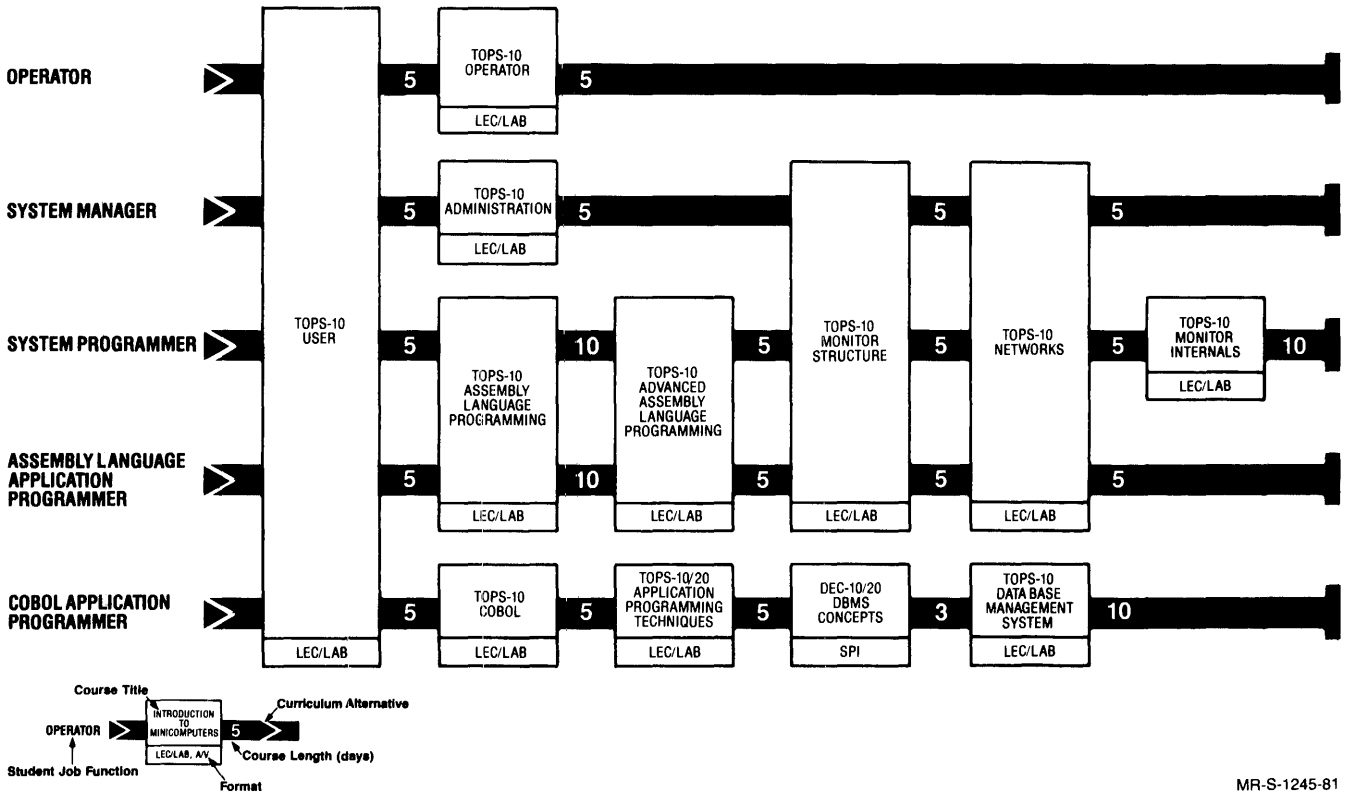


Figure 11–1
TOPS–10 Courses

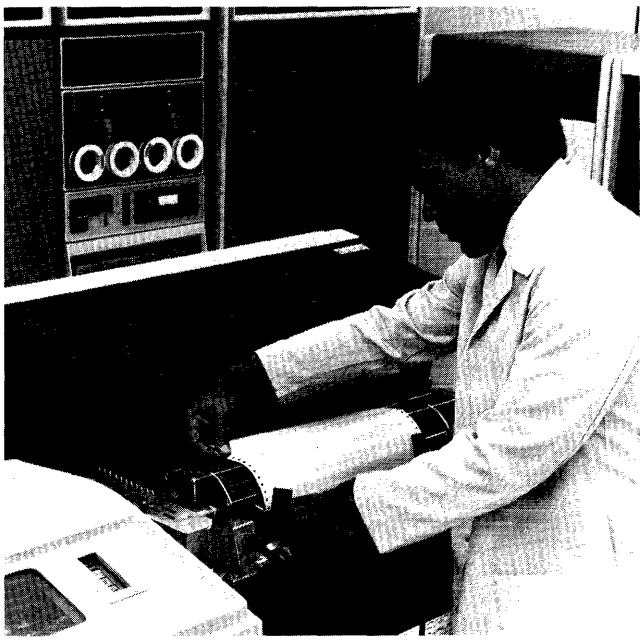
TOPS–10 Monitor Structure teaches the structure and the basic components and functions of the monitor with emphasis on the batch system, command processor, and the monitor's allocation and manipulation of system resources.

TOPS–10 Monitor Internals covers the monitor in depth, beyond the structural level, for the programmer responsible for monitor patching and problem detection. Module organization, monitor data base, and internal algorithms are stressed.

TOPS–10 Data Base Management System (DBMS) gives managers, systems programmers, and applications programmers both the conceptual understanding and programming skills necessary to implement DBMS systems.

Hardware Services

DIGITAL's Field Service organization offers a range of post-warranty hardware maintenance services. Over 10,000 Field Service personnel in more than 400 locations worldwide are ready to provide the support needed for continuous productivity.



DECservice is DIGITAL's most comprehensive on-site maintenance plan. It is designed for customers who require uninterrupted system performance. DECservice includes:

- Four-hour response time
- Continuous-effort remedial service
- Priority-problem escalation
- Scheduled preventive maintenance
- Parts, labor, and materials
- Installation of the latest engineering change orders
- Assigned service representative
- Comprehensive site management guide

Field Service's other on-site maintenance plan is called Basic Service. Designed for customers who do not require a fixed response time and continuous remedial efforts, Basic Service provides:

- Next day response
- Remedial service during coverage hours
- Priority-problem escalation
- Preventive maintenance during coverage hours
- Parts, labor, and materials
- Installation of the latest engineering change orders
- Assigned service representative
- Comprehensive site management guide

Although standard coverage for both on-site service plans is eight hours a day, five days a week, customers can opt to extend their service cover age to 12, 16, or 24 hours, including weekends and holidays.

Hardware maintenance on a per-call time-and-materials basis is also available.

More information is available on DIGITAL's Field Service hardware maintenance offerings from your local DIGITAL sales representative.

Customer Financing

To simplify the financial considerations involved in acquiring a new computer, DIGITAL provides leasing and financial counseling. The Customer Finance Department can help customers acquire a DIGITAL system through a lease, conditional sale, or similar financing agreement, rather than an outright cash purchase.

For commercial businesses or private organizations, DIGITAL has developed a program known as DIGITAL Leasing with the U.S. Leasing Corporation of San Francisco. DIGITAL Leasing, a division of U.S. Leasing, is committed solely to financing DIGITAL computers. Representatives are located in or near many of the DIGITAL District Sales Offices.

Federal, state, and local government agencies have special contractual needs and, in some cases, can benefit from special tax privileges. For example, a state or municipal agency qualifies for special interest rates on Conditional Sales Agreements, rates that are significantly lower than those charged to commercial customers. The interest income is free from federal and, in some cases, state income taxes.

The following financing is available: Full Payout Lease, Conditional Sales Agreement, and Federal Government Lease to Ownership Agreement.

- Full Payout Lease — This is used primarily by commercial customers. It involves a noncancelable three- to five-year term, usually with a 10 percent purchase option at the end. No down payment is required, and title remains with the lessor. Flexible lease payment schedules can be tailored to specific needs.
- Conditional Sales Agreement — This type of financing is used primarily by state and local governments. It involves a noncancelable one- to five-year term. Title passes to the customer but DIGITAL retains a security interest. The customer owns the equipment free and clear at the end of the term. Fiscal funding provisions are available for state and local governments.
- Federal Government Lease to Ownership Agreement — This is available only to approved federal government agencies. It involves a one- to five-year term, cancelable at the end of each fiscal year for nonappropriation of funds. Ownership passes to the customer at the end of the term.

DIGITAL's Customer Financing group can provide financial counseling to help you decide which arrangement is best for you. For more information, contact your local DIGITAL sales office.

Accessories and Supplies Group

DIGITAL's Accessories and Supplies Group (A&SG) maintains Accessories and Supplies Centers (ASCs) and offers direct factory ordering, the Direct Sales Catalog, and worldwide support for their products.

ASCs are full-service centers established to fulfill the needs of DIGITAL customers in major metropolitan areas. The ASCs hold a local inventory of the most requested accessories, supplies, documentation, and add-on products for fast delivery. Full order processing capability provides access to A&SG's central inventory in Nashua, New Hampshire. ASCs provide first-class service and convenience to DIGITAL's customers.

A&SG maintains a toll-free telephone number for customers to use when ordering accessories and supplies. The majority of products are shipped within 48 hours of receipt of order.

A&SG's Direct Sales Catalog offers a broad range of computer accessory and supply items. These include small computer systems and their complementary options, accessories, and operating supplies. The Direct Sales Catalog also features some DIGITAL hardware options such as DECwriters, microcomputers, and their associated options. Hardware and software documentation is also offered.

A&SG has a team of worldwide specialists and business managers to support sales. This sales force is located in the United States, Europe, and the General International Area (GIA).

Computer Supplies

DIGITAL's Computer Supplies group maintains a complete line of supplies specifically designed for use with DIGITAL systems. These items facilitate reliable and efficient system operation and include:

- A family of magnetic media such as disk cartridges, disk packs, and floppy diskettes
- Self-contained disk cartridge cleaners for fast and efficient cleaning of front- or top-loading magnetic disk cartridges
- Word processing supplies, such as nylon and mylar ribbons, a selection of 11 print wheels, and filter screens for video terminals
- Ribbons for DIGITAL's DECwriter and DECprinter terminals

The Computer Supplies group also offers computer cabinetry for maintaining supplies and protecting magnetic media not in use. Cabinet interiors can be customized with various options to meet individual needs. Options can be conveniently rearranged or changed at any time to adapt to future requirements. Also available are paper baskets, work shelves, terminal tables, tape racks, paper tape trays, and multipurpose binders.

Relying on DIGITAL for computing needs means a savings in time, money, and paperwork. Contact your sales rep for further information.

Customer Spares

Customer Spares is dedicated to supporting customers who maintain their own computers. Customer Spares is organized into three distinct businesses: self-maintenance products (hardware and documentation), system accessories, and low-volume LSI–11 products. System accessories include products geared to the hardware builder. They allow easy expansion and re-configuration of DIGITAL systems and options.

Customer Spares sells modules, subassemblies, components, tools, and test equipment. Related services involve providing assistance in selecting the proper parts and expediting delivery during emergency situations.

DECUS

DECUS, Digital Equipment Computer Users Society,

is one of the largest and most active user groups in the computer industry. It is a not-for-profit association supported and administered by DIGITAL, but actively controlled by individuals who have purchased, leased, ordered, or used a DIGITAL computer or who have a bona fide interest in DECUS. Membership is free and voluntary.

The goals of DECUS are to:

- Advance the art of computation through mutual education and the exchange of ideas and information
- Establish standards and provide channels that ease the exchange of computer programs
- Provide feedback to DIGITAL on hardware and software needs
- Advance the effective use of DIGITAL computers, peripherals, and software by promoting the interchange of information

DECUS headquarters, located in Marlborough, Massachusetts, administers all international policies and activities. DECUS is subdivided into four chapters:

DECUS AUSTRALIAN CHAPTER (Australia, Brunei, New Zealand, Malaysia, Singapore, Indonesia, PNG):

DECUS Australia
P.O. Box 384
Chatswood
NSW 2067
Australia

DECUS EUROPEAN HEADQUARTERS (Europe, Middle East, North Africa, Eastern Europe):

DECUS Europe
P.O. Box 510
12, Av. Des Morgines
CH–1213 Petit-Lancy/GE
Switzerland

DECUS CANADIAN CHAPTER:

DECUS Canada
P.O. Box 13000
Kanata, Ontario Canada

DECUS U.S. CHAPTER (for U.S. and all others):

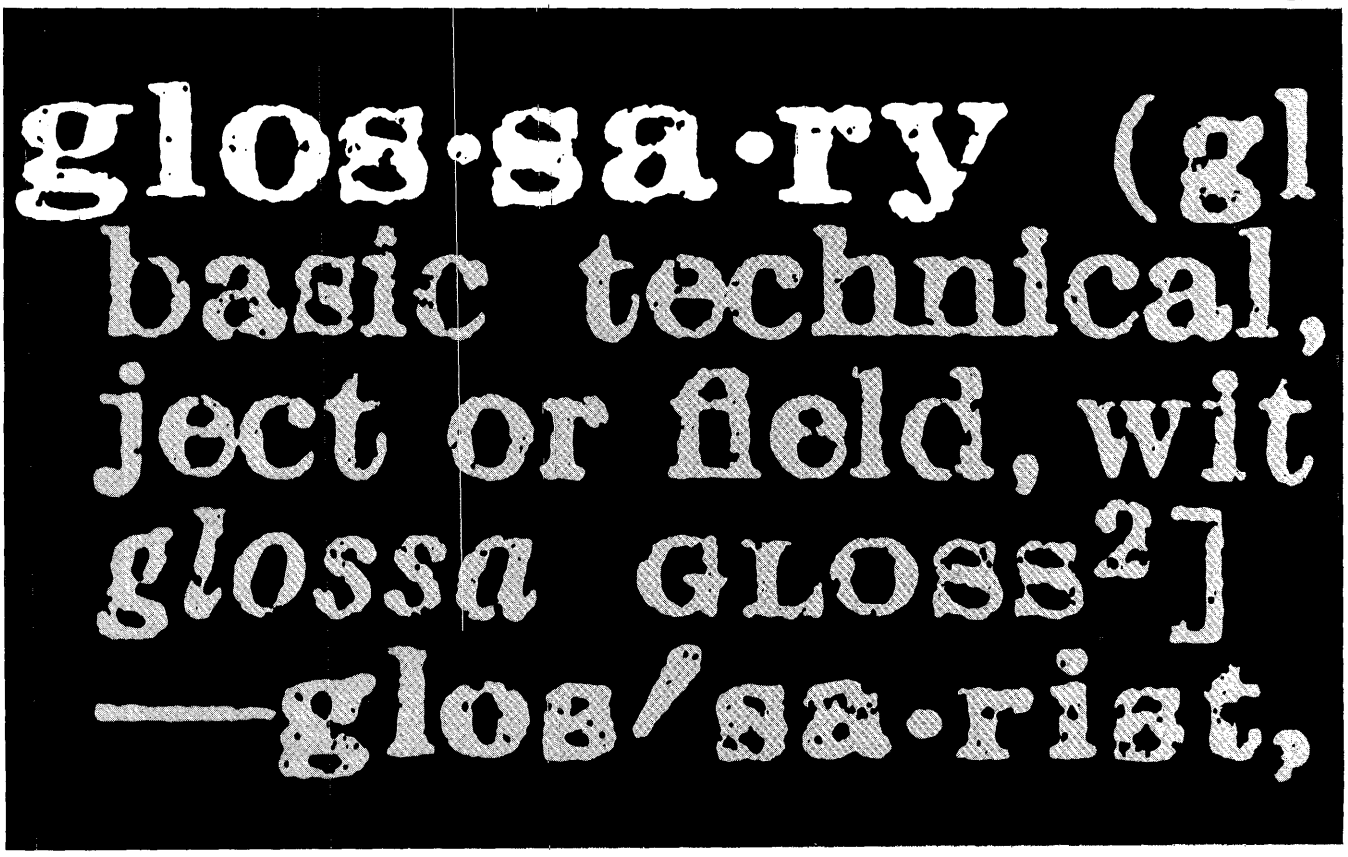
DECUS International Headquarters
Digital Equipment Corporation
MR2–3/E55
One Iron Way
Marlborough, MA 01752 U.S.A.

To further the goals of the society, DECUS serves its members by holding symposia; maintaining a program library; publishing an association newsletter, technical newsletters and books; and supporting a number of special user groups for special interests and locations.

- Symposia: These are regularly scheduled meetings held in each of the four chapters. They provide a forum in which users of DIGITAL products can meet with other users and with DIGITAL management, engineering, and support personnel. Symposia give users an opportunity to participate in DIGITAL product workshops and product planning feedback sessions. Many of the technical papers and presentations from each symposium are published as a book, the DECUS Proceedings. Copies of the Proceedings are supplied to all symposia attendees and may be purchased by any DECUS member.
- Program Library: A major activity of DECUS is the Program Library. It contains over 1700 active software packages written and submitted by users. A wide range of software is offered including languages, editors, numerical functions, utilities, display routines, games, and other types of application software. Library catalogs are available for the DECsystem-10. Catalogs are updated yearly and contain program descriptions and ordering information. The programs are available at nominal service charges that cover the cost of reproduction and media.
- Association Newsletter: Each DECUS chapter publishes and distributes to all chapter members a newsletter of general DECUS news.
- Special Interest Groups: The focus of these groups is on operating systems, languages, processors, and applications. Local User Groups, National User Groups, and Regional User Groups, which are formed basically by geographical proximity, may also share common specific interests. Many of these subgroups also publish newsletters.

You can obtain a membership form for DECUS by contacting your sales rep or the appropriate chapter office.

The Glossary



Absolute virtual address A fixed location in user virtual address space that cannot be relocated by the software. However, it can be translated to a physical address by the hardware. For example, the high-speed accumulators on TOPS–10 occupy locations 0 through 17(8) in the user's virtual address space. All modules that reference the accumulators must reference these locations. Thus, the addresses 0 through 17(8) are absolute virtual addresses.

Access date The date on which a file on disk was last read or written. If a file has not been read or written since it was created, the creation date and the access date are the same. The access date is kept in the Retrieval Information Block (RIB) for the file.

Access privileges Attributes of a file that specify the class of users allowed to access the file and the type of access they are allowed.

Access table The table stored in the monitor that reflects the status of all files open for reading or writing in addition to the status of those files recently closed. This information is kept in the monitor in order to decrease the time needed to access the files.

Accumulator One of the registers and associated equipment in the arithmetic unit in which data can be placed while it is being examined or manipulated (i.e., the 16 high-speed registers at address locations 0 through 17(8). The index registers are a subset of the accumulators 1 through 17.

Address

1. An identification represented by a name, label, or number for a register, a location in storage, or any other data source or destination in memory or on an addressable storage device.
2. The part of an instruction that specifies the location of an operand of the instruction.

Address mapping The assignment of user virtual address space to the physical address space in computer memory. This is automatically performed by the TOPS–10 monitor and is completely invisible to user programs.

Alphanumeric The characters which include the letters of the alphabet (A through Z), and the numerals (0 through 9).

Arithmetic unit The portion of the central processing unit in which arithmetic and logical operations are performed.

ASCII code American Standard Code for Information Interchange. A 7-bit code in which textual information is recorded. The ASCII code can represent 128 distinct characters. These characters are the upper and lower case letters, numbers, common punctuation marks, and special control characters.

Assembly language The machine-oriented symbolic programming language specific to a given computing system. The assembly language for TOPS–10 is MACRO.

Asynchronous

1. Pertaining to the procedure by which the hardware can begin a second operation before the first operation is completed.
2. Pertaining to the method of data transmission in which each character is sent with its own synchronizing information and with no fixed time between consecutive characters.

Baud A unit of signaling speed equal to the number of discrete conditions or signal events per second.

Binary code A code that uses two distinct characters. Usually these characters are 0 and 1.

Bit A binary digit (i.e., 0 or 1) that usually refers to the smallest unit of information storage, which can be on or off (i.e., 1 or 0). A word on the TOPS–10 has 36 bits.

Block A set of records, words, characters, or digits handled as a unit. On the TOPS–10, a 128-word unit of disk storage allocated by hardware and software; 128 words are always written, adding zeroes as necessary, although fewer than 128 words can be read.

Bootstrap A routine or device designed to bring itself into a desired state by means of its own action, e.g., a machine routine whose first instructions are sufficient to bring the rest of itself into the computer from an input device.

Breakpoint A location at which program operation is suspended in order to examine partial results. Breakpoints are used in the debugging process.

Buffer A device or area used temporarily to hold information being transmitted between two processes, such as external and internal storage devices or I/O devices and internal high-speed storage. A buffer is often a special register or a designated area of internal storage.

Buffer pointer A position indicator that is located between two characters in an editing buffer, before the first character in the buffer, or after the last character in the buffer.

Byte Any contiguous set of bits within a word.

Call To transfer control to a specified closed subroutine.

Calling sequence A specified arrangement of instructions, pointers, and data necessary to pass

parameters and control to and return from a given subroutine.

Card On the TOPS–10, a punched card with 80 vertical columns, each containing 12 vertical rows. A card can also be a unit of computer circuitry.

Card column One of the vertical lines of 12 punching positions on a punched card.

Card field A fixed number of consecutive card columns assigned to a unit of information.

Card hopper The tray on a card processing machine that holds the cards to be processed and makes them available to the card feed mechanism.

Card row One of the horizontal lines of punching positions on a punched card. A row is 80 columns long.

Card stacker The tray on a card processing machine that receives processed cards.

Central processing unit (CPU) The portion of the computer that contains the arithmetic and logical facilities, control circuits, and basic I/O and memory interfaces. There can be more than one CPU in a computing system.

Central site The location of the central computer; used in conjunction with remote communications to mean the location of the TOPS–10 central processor, as distinguished from the location of the remote station.

Channel

1. A path along which signals can be sent; e.g., output channel.
2. A portion of the TOPS–10 which can overlap I/O transmission while computations proceed simultaneously (e.g., data channel).

Character One symbol of a set of elementary symbols, such as those corresponding to the keys on a typewriter. The symbols usually include the decimal digits 0 through 9, the letters A through Z, punctuation marks, operation symbols, and any other special symbols which a computer may read, store, or write.

Clear To erase the contents of a location, a block of memory, or a mass storage device by replacing the contents with blanks or zeroes.

Command The part of an instruction that causes the computer to execute a specified operation.

Command List Specifies the area in your area to be read or written when performing dump I/O.

Communication link The physical means of connecting one device to another for the purpose of transmitting and receiving data.

Communication Among Jobs In TOPS–20, data transmitted and received from and by accomplished by Inter-Process Communication Facility.

Computer operator A person who manipulates the controls of a computer and performs all operational functions that are required in a computing system, such as adjusting parameters which affect the operation of the system, loading a tape transport, placing cards in the input hopper, and removing listings from the line printer.

Computer program A series of instructions or statements prepared to achieve a specific result and intended for execution by a computer. A program can be in either the binary form in which it can be directly executed by a computer, or a symbolic form which must be compiled or assembled before it can be executed.

Concatenation Joining two strings of characters to produce a longer string, often used to create symbols in macro definitions.

Console The part of a computer used by the operator to determine the status of, and to control the operation of, the computer; also informally used to refer to the user's terminal.

Context switching Saving sufficient hardware and software information of a process so that it may be continued at a later time, and the restoring of similar information relevant to another process. A common example of context switching is the temporary suspension of a user program so that the monitor can execute a function.

Control character A character whose purpose is to control an action, such as line spacing on the line printer, rather than to pass data to a program. An ASCII control character has an octal representation of 0–37. It is typed by holding down the CTRL key on the terminal while striking a character key. It can be punched on a card via the multipunch key.

Controller The device or portion of a device which controls the operation of connected units. Some controllers can initiate simultaneous positioning commands to some of its units and can then perform a transfer for one of its units.

Counter A device, such as a register or storage location, used to represent the number of occurrences of a certain event. See program counter.

CPU See central processing unit.

Create To open, write, and close a file for the first time. Only one user at a time can create a file with a

given name and extension in the same directory or subdirectory of a file structure.

Customer A customer of Digital Equipment Corporation who has purchased a TOPS–10, as distinguished from a user at a terminal who may be purchasing time from a customer.

Cylinder The hardware-defined region of consecutive logical disk blocks which can be read or written without repositioning. A cylinder usually consists of tracks in the same physical position on different disk surfaces.

Data A general term used to denote any or all information (facts, numbers, letters, and symbols that refer to or describe an object, idea, condition, or situation). It represents basic elements of information which can be processed by a computer.

Data channel The device which passes data between the memory system and a controller.

DDT The Dynamic Debugging Technique program; used for on-line checkout, testing, examination, modification, and program composition of object programs.

Deadlock Two or more jobs waiting for each other to complete use of a resource, but neither of the jobs can obtain a lock on the resource it needs for completion.

Debug To detect, locate, and correct any mistakes in a computer program.

DECtape A convenient, pocket-sized reel of random access magnetic tape developed by Digital Equipment Corporation. A standard reel consists of 578 (decimal) blocks, each capable of storing 128 (decimal) words of data.

Default directory The directory in which the Monitor searches when a directory specification has not been given by the user. Typically, this is the UFD (user-file directory) corresponding to the user's logged-in project-programmer number but it may be another UFD or a SFD (sub-file directory).

Demand paging The operation in which all pages of a program are not resident in memory during execution. References to nonresident pages initiate the actions of moving in additional pages or replacing inactive pages.

Device routines Routines that perform I/O for specific hardware devices. They usually translate logical block numbers to physical addresses for those devices that associate addresses with data. These routines also handle error recovery and ensure ease of programming through device independence.

Diagnostic The detection and isolation of a hardware malfunction or bug. A program which tests the hardware and isolates any faults.

Digit A symbol that represents one of the nonnegative integers smaller than the base of the system. For example, in the decimal system, a digit is one of the characters from 0 to 9.

Direct address An address that specifies the location of an operand. Contrast with indirect address.

Directory A file which contains the names and pointers to other files on the device. The MFD, UFDs, and SFDs are directory files. The MFD is the directory containing all the UFDs. The UFD is the directory containing the files existing in a given project-programmer number area. The SFD is a directory pointed to by a UFD or a higher-level SFD. The SFDs exist as files under the UFD. The DIRECT monitor command lists a directory.

Directory device A storage retrieval device, such as disk or DECtape, which contains information describing the names of files and the layout of stored data (programs and other files). A directory device is randomly accessible.

Directory path (path) The ordered list of directory names, starting with a UFD name, which uniquely specifies a directory without regard to a file structure. A file structure name, a path, and a filename and extension are needed to uniquely identify a file in the system.

Disk A form of mass storage device in which information is stored on rotating magnetic platters. On TOPS–10, a disk is a directory device.

Disk address A logical or relative address. References to disk addresses do not refer to any physical addressing scheme. The basic addressable unit is a 200 (octal) 36-bit word block.

Double precision The use of two computer words to represent a number.

Dump A listing of variables and their values, or a listing of the values of locations in memory.

Echoing A method of data transmission in which the received data is returned to the sending end.

Edit To modify the content or format of a program or data file (e.g., to insert or delete characters).

Effective address The actual address used; that is, the specified address, as modified by any indexing or indirect addressing rules.

Enqueue/Dequeue A facility that ensures that resources (e.g., files) are shared correctly.

Entry point A point in a subroutine to which control is transferred when that subroutine is called.

Error Interception The activity of the monitor in an error condition. When an error occurs, the monitor intercepts control of the program, examines location .JBINT, and transfers control to an error intercepting routine.

Execute To interpret an instruction or command and perform the indicated operation(s).

Executive mode A central processor mode characterized by the lack of memory protection and relocation and by the normal execution of all defined operation codes.

External symbol A global symbol which is referenced in one module but defined in another module. The EXTERN statement in MACRO–10 is used to declare a symbol external. A subroutine name referenced in a CALL statement in a FORTRAN module is automatically declared external.

File An ordered collection of characters or 36-bit words containing computer instructions and/or data. A file is stored on a device, such as disk or magnetic tape, and can be of any length, limited only by the available space on the device and the user's maximum space allotment on that device. A file is uniquely identified in the system by a file structure name or directory name, a directory path, and a filename and extension.

File directory See directory.

File-structure device A device on which data is given names and arranged into files. The device also contains directories of these names. It is usually synonymous with directory device.

Filename One to six alphanumeric characters chosen by the user to identify a file.

Filename extension One to three alphanumeric characters usually chosen by the program to describe the class of information in a file. The extension is separated from the filename by a period.

File specification A list of identifiers which uniquely specifies a particular file. A complete file specification consists of: the name of the device on which the file is stored, the name of the file including its extension, and the name of the directory in which the file is contained. File specifications are ignored for nonfile-oriented devices, such as cards and paper tape.

File specification area The area of core in which SCAN stores the result of scanning the user's file specification. This instructs WILD as to the files to select.

File status The status of a file set in the file status word.

File structure The logical arrangement of blocks (which are normally 128 words long) on one or more physical I/O device units of the same type to form a collection of named files. It is the smallest removable component in the file system and has its own MFD.

File structure abbreviation An abbreviation of one or more file structures. This refers to all those structures in the ALL search list whose names match the abbreviation. For example, if there were structures "PRIV."and "PACK:", "P" would refer to both structures, but "PR:" would mean just "PRIV:".

File structure owner The user whose project-programmer number is associated with the file structure in the administrative file STRLST.SYS. The REACT program is used to enter or delete this project-programmer number or any of the other information that is contained in an STRLST.SYS entry.

Flag An indicator that signals the occurrence of some condition, such as the end of a word.

Global symbol A symbol that is accessible to modules other than the one in which it is defined. The value of a global symbol is placed in the loader's global symbol table when the module containing the symbol definition is loaded.

Group A contiguous set of disk clusters allocated as a single unit of storage and described by a single retrieval pointer.

Half word A contiguous sequence of bits or characters which comprises half of a computer word and may be addressed as a unit. On the TOPS–10, bits 0 through 17 comprise the left half word and bits 18 through 35, the right half word. Each half word is 18 bits long.

Hardware Physical equipment of the computer (e.g., magnetic, mechanical, and electronic devices), as contrasted with the computer program (software) or method of use.

High segment That portion of the user's addressing space, usually beginning at virtual address 400000, which generally is used to contain pure code that can be shared by other users. This segment is usually write-protected in order to protect its contents. The user can place information into a high segment with the TWOSEG pseudo-op in MACRO–10. Higher-level languages, such as COBOL and FORTRAN, also have provisions for loading pure code in the high segment.

Home block The block written twice on every unit which identifies the file structure the unit belongs to and its position on the file structure. This block specifies all the parameters of the file structure along with

the location of the MFD. The home block appears in the HOME.SYS file.

Host Site See central site.

I/O An abbreviation for input or output, or both; pertaining to all equipment and activity that transfers information into or out of a computer.

Idle time That part of uptime in which no job could run because all jobs were HALTed or waiting for external action such as I/O.

Immediate mode addressing The interpretation of certain instructions in which the effective address of the instruction is used as the value of an operand (rather than the address of an operand).

Indexed address An address that is formed by adding the content of an index register to the content of an address field prior to or during the execution of a computer instruction.

Index register A register whose contents may be added to the operand address prior to or during the execution of a computer instruction. On the TOPS–10, accumulators 1 through 17 (octal) may be used as index registers (accumulator 0 may not be used as one).

Indirect address An address which indicates a storage location where the address of the referenced operand (or another indirect address) is to be found. Contrast with direct address.

Initialize To set counters, switches, or addresses to zero or other starting values at prescribed points in the execution of a computer routine, particularly in preparation for reexecution of a sequence of code.

Input

1. Pertaining to a device, process, or channel involved in the acquisition of data.
2. Information that is read by a computer.

Instruction A bit pattern which, when interpreted by the computer, directs the computer to execute a specific operation. An instruction generally contains the values or locations of its operands.

Interleaving The process of configuring the memory addressing so that consecutive addresses are not stored in the same memory module. This allows the possibility of increasing memory speed by overlapping part of the operation of different memory modules.

Internal storage Addressable high speed storage directly controlled by the central processing unit.

Interrupt A signal which, when activated, causes a transfer of control to a specific location in memory thereby breaking the normal flow of control of the routine being executed. An interrupt is caused by an external event such as a done condition in a peripheral. It is distinguished from a trap which is caused by the execution of a processor instruction.

IPCF The Inter-Process Communications Facility, which allows communication among jobs and system processes.

Job The entire sequence of steps from beginning to end, that the user initiates from his interactive terminal or batch control file or that the operator initiates from his operator's console. Thus, it is a specific group of steps presented as a unit of work for the computer. A job usually includes all necessary computer programs, files, linkages and instructions to the operating system.

Job Data Area (JOBDAT) The first 140 octal locations of a user's virtual address space. This area provides storage for certain data items used by both the Monitor and the user's program.

Label A symbolic name used to identify a statement or an item of data in a program.

Leader A blank section of tape at the beginning of a reel of magnetic tape or the beginning or end of a stack of paper tape.

Library A file containing one or more relocatable binary modules which may be loaded in Library Search Mode. MAKLIB is a system utility program which enables users to merge and edit a collection of relocatable binary modules into a library file. PIP can also be used to merge relocatable binary modules into a library, but it has no facilities for editing libraries.

Library search mode The mode in which a module (one of many in a library) is loaded only if one or more of its declared entry points satisfy an unresolved global request.

Library search symbol (entry symbol) A list of symbols that are matched against unresolved symbols in order to load the appropriate modules. This list is used only in library search mode. A library search symbol is defined by an ENTRY statement in MACRO–10.

Line A string of characters terminated with a vertical tab, form feed, or line feed. The terminator belongs to the line that it terminates.

Line feed

1. The operation that prepares for the next character to be printed or displayed at the same (current) position on the next line on a terminal or line printer.
2. The ASCII character with the octal code 012.

Line printer An electro-mechanical computer peripheral which accepts a line of characters from the computer at a high speed and then prints the entire line in one operation.

Link To combine independently-translated modules into one module in which all relocation of addresses has been performed relative to that module and all external references to symbols have been resolved based on the definition of internal symbols.

Local peripherals The I/O devices and other data processing equipment and memory, excluding the central processor and memory, located at the central site.

Lock An association between a job and a resource.

Locked job A job in core that is never a candidate for swapping or shuffling.

Logical device name An alphanumeric name you choose to represent a physical device. This name can be used synonymously with the physical device name in all references to the device. Logical device names allow device independence in that the most convenient physical device can then be associated with the logical name at run time. Logical names take precedence over physical names. With the exception of disks, only one logical name can be associated with a physical name.

Logical record A collection of related items stored together. It is possible to have:

- Several logical records stored in a single physical record.
- Each logical record stored in a single physical record.
- Each logical record occupy one or more physical records.
- Logical records span several physical records, and at the same time, have more than one logical record in a single physical record.

LOGIN The system program by which the system users gain access to the computing system.

Low segment The segment of user virtual address space beginning at zero. It contains the Job Data Area and I/O buffers. The length of the low segment is stored in location .JBREL of the Job Data Area. When writing two-segment programs, it is advisable to place data locations and impure code in the low segment.

MACRO The symbolic assembly program on the TOPS–10.

Macro A body of text which is substituted for its name whenever its name is invoked.

Magnetic tape A tape with a magnetic surface on which data can be stored by magnetizing selective portions of the surface.

Mask

1. A combination of bits that is used to control the retention or elimination of portions of any word, character, or byte in memory.
2. On half-duplex circuits, the characters typed on the terminal to make the password unreadable.

Master file directory (MFD) The file created at refresh time which contains the name of all user file directories including itself.

Memory protection A scheme for preventing read and/or write access to certain areas of storage.

Metering A technique used to perform performance analysis.

Mnemonic symbol A symbolic representation for a computer instruction or other numeric item. All defined monitor symbols are listed in UUOSYM.MAC.

Mode One of ten modes that can be used when performing I/O.

MONGEN The monitor generator dialogue program that enables the system programmer to define the hardware configuration of his individual installation and the set of software options that he wishes to select for his system.

MONGEN time The time at which the monitor software configuration is being defined or changed. The monitor must then be reloaded with the loader.

Monitor

1. The collection of programs which schedules and controls the operation of user and system programs, performs overlapped I/O, provides context switching, and allocates resources so that the computer's time is efficiently used.
2. The operating system.

Monitor command An instruction to the monitor to perform an operation.

Mount count The count of the number of jobs which have a file structure in their active or passive search lists (plus 1 if the structure is in the system search list).

Multiprocessing Simultaneous execution of two or more computer programs (i.e., processes) by two or more processors.

Multiprogramming A technique that allows scheduling in such a way that more than one job is in an executable state at any one time. TOPS–10 is a multiprogramming operating system in which there are two or more independent instruction streams that are

simultaneously active but are not necessarily simultaneously executed.

Named file A named ordered collection of 36-bit words (instructions and/or data) whose length is limited only by the available space on the device and the user's maximum space allotment on that device.

Nondirectory device A device such as a magnetic tape or paper tape, on the TOPS–10 which does not contain a file describing the names and layout of data files.

Octal

1. Pertaining to a characteristic or property in which there are eight possibilities.
2. Pertaining to the number system with a radix of eight.

Operand

1. The data which is accessed when an operation (either a machine instruction or a higher level operation) is executed.
2. The symbolic expression representing that data or the location in which that data is stored, for example, the input data or arguments of a pseudo-op or macro instruction.

Operating system The collection of programs that administer the operation of the computing system by scheduling and controlling the operation of user and system programs, performing I/O and various utility functions, and allocating resources for efficient use of the hardware. See also monitor.

OPSER The OPerator SERvice program that facilitates multiple job control from a single terminal by allowing the operator or user to initiate several jobs from his terminal.

Output

1. Pertaining to a device, process, or channel involved in an output process (i.e., the process of transferring data from memory to a peripheral device).
2. The data that has been transferred from memory to a medium readable by a person.

Packet A group of words transmitted via IPCF to and from jobs and/or system processes.

Pack ID A 6-character SIXBIT name or number used to uniquely identify a disk pack.

Page

1. Any number of lines terminated with a form feed character.
2. The smallest mappable unit of core storage. On the KL10 processor, a page is 512 continuous words in core starting on boundaries which are even multiples of 512. It is also the smallest allocatable unit of memory. KL10 operations allow programs to be composed of up to 512 pages scattered within core.
3. To selectively remove parts of a user's program from main memory.

Paper tape A tape on which data is represented by specific patterns of punched holes.

Parameter A variable that is given a constant value for a specific purpose or process, for example, an input argument to a subroutine or command, or a value specifically assigned to a symbol in an assembly in order to control exactly what code is assembled.

Parity bit A binary digit appended to a group of bits to make the sum of all the bits always odd (for odd parity) or always even (for even parity).

Parity check A check that tests whether the number of ones or zeros in an array of binary digits is correct. This check helps ensure that the data read has not been unintentionally altered.

Password The character string assigned to a user, known only by the user, the installation administration, and the monitor system. The password is used to verify that a user is entitled to run a job under a specific user number (project-programmer number).

PC See program counter.

Peripheral equipment Any unit of equipment, distinct from the central processing unit, the console, and the memory, that can provide input to, or accept output from, the computer.

Physical address space A set of separate memory locations where information can actually be stored (i.e., MOS memory) for the purpose of program execution. Virtual memory addresses may be mapped, relocated, or translated to produce a final memory address which is sent over the memory bus to hardware memory units. This final address is the physical address and is 22 bits long on the TOPS–10.

PI See priority interrupt.

PID A process' packet identifier when using the Inter-Process Communication Facility.

Pointer

1. A location or register containing an address rather than data. A pointer may be used in indirect addressing or in indexing.
2. An instruction indicating the address, position, and length of a byte of information (i.e., a byte pointer).

Priority interrupt An interrupt that usurps control of the computer from the program or monitor and jumps

to an interrupt service routine if its priority is higher than the interrupt currently being serviced.

Process A collection of segments that perform a particular task. A hardware state is associated with a process: a virtual address space, a processor state, a stack, etc.

Program

1. The complete plan for the solution of a problem, more specifically the complete sequence of machine instructions and routines necessary to solve a problem.
2. A collection of routines which have been linked and loaded to produce a saved file or a core image. These routines typically consist of a main program and a set of subroutines, some of which may have come from a library.

Program counter (PC) A register that contains the address from which the next instruction to be executed is fetched. At the beginning of each instruction on the TOPS–10, the PC normally contains an address one greater than the location of the previous instruction.

Programmed operators Instructions which, instead of performing a hardware operation, cause a jump into the monitor system or the user area at a predetermined point and perform a software operation. The monitor (or special user code) interprets these entries as commands from the user program to perform specified operations.

Program origin The location assigned by LINK to relocatable zero of a program.

Program trap One of the nonhardwired operation codes which, when decoded by the processor, causes the next instruction to be executed from a specified address.

Project-programmer number Two octal numbers, separated by commas, which, when considered as a unit, identify the user and his file storage area on a file structure.

Protected location

1. A storage location which cannot be accessed in a certain context. For example, a write-protected location cannot be written into.
2. A storage location reserved for special purposes in which data cannot be stored without undergoing a screening procedure to establish suitability for storage therein.

Protection address The maximum relative address that the user can reference.

Protection Code Each file has a protection code that indicates who may and may not access the file.

Pseudo-op An operation that is not part of the computer's operation repertoire as realized by hardware; hence, an extension of the set of machine operations. In MACRO, pseudo-ops are directions for assembly operations.

Public disk pack A disk pack belonging to the storage pool and whose storage is available to all users who have quotas on it.

Quantum time The run time given to each job when it is assigned to run.

Queue A list of items waiting to be scheduled or processed according to system, operator, or user-assigned priorities. Examples: Batch input queue, spooling queues, monitor scheduling queues.

Random access A process having the characteristic that the access time is effectively independent of the location of the data.

Read Input data from a file.

Record A collection of adjacent related items of data treated as a logical unit.

Record gap An area on a data medium between consecutive records. It is sometimes used to indicate the end of a block or record.

Reentrant program A program consisting of sharable code which can have several simultaneously independent users.

Relative address The address before hardware or software relocation is added.

Relocatable address An address within a module which is specified as an offset from the first location in that module.

Relocate

1. To move a routine from one portion of storage to another and to adjust the necessary address references so that the routine can be executed in its new location.
2. To convert a relocatable binary module to an absolute binary module.

Remote access Pertaining to communication with a data processing facility by one or more stations that are distant from that facility.

Remote peripherals The I/O devices and other data processing equipment, located at the site of the remote Batch terminal.

Resource A entity within the system. The actual definition of a resource is defined by the job using that resource, not by the system.

Response time The time between the generation of an inquiry or request and the receipt of the response or the accomplishment of the requested action.

Routine A set of instructions and data for performing one or more specific functions.

Run To transfer a save file from a device into memory and to begin execution.

Secondary storage Low speed magnetic storage such as disks or drums.

Sector A physical portion of a mass storage device.

Segment An absolute Control Section. A logical collection of data, either program data or code, that is the building block of a program. The monitor keeps a segment in core and/or on the swapping device.

Service routine a routine in general support of the operation of a computer (e.g., an input-output, diagnostic, or monitoring routine).

SFD (sub-file directory) A directory pointed to by a UFD or a higher-level SFD. Each user has a UFD. Within that, he may have as many SFDs as he wishes. He may nest the SFDs up to installation maximum. The installation maximum cannot be greater than 5 SFDs deep.

Sharable segment A (high) segment which can be used by several users at a time.

Sharer's Group A subset of those jobs desiring shared ownership of a particular resource. Refer to Chapter 9.

Simultaneous Update Allowing more than one cooperating job to update a file.

Single access The status of a file structure that allows only one particular job to access the file structure. This job is the one whose project number matches the project number of the owner of the file structure.

SIXBIT code A 6-bit code in which textual information is recorded. It is a compressed form of the ASCII character set, and thus not all of the characters in ASCII are available in SIXBIT, notably the nonprinting characters and the lower case letters are omitted. The range of SIXBIT code is 00 to 77 (octal) which is equal to 40 through 137 (octal) in ASCII.

Skip The process by which an instruction, macro or subroutine causes control to bypass one instruction and proceed to the next instruction.

Software Interrrupt System The interruption of the sequential flow of program execution under a variety of conditions.

Spooling The technique by which output to slow-speed devices is placed into queues on faster devices (such as disk) to await transmission to the slower devices; this allows more efficient use of the particular device, memory, and the central processor unit.

Storage Allocation Table A file reflecting the status of every addressable block on the disk.

String A set of contiguous items of a similar type. Generally strings are sequences, of variable or arbitrary length; of bits, digits, or characters.

Structure A file structure.

Sub-File Directory A continued SFD.

Subroutine A routine designed to be used by other routines to accomplish a specific task.

Swapping

1. The technique in multiprogramming of running more jobs than there is physical memory for, by storing some of the jobs on secondary storage when they are not executing.
2. The movement, by the monitor, of user programs between core and secondary storage.

Swapping class A category of swapping units distinguished from other categories of swapping units according to speed.

Swapping device Secondary storage that is suitable for swapping, usually a high-speed disk or drum.

Symbol Any identifier used to represent a value that may or may not be known at the time of its original use in a source language program. Symbols appear in source language statements as lables, addresses, operators, and operands.

Symbolic address An address used to specify a storage location in the context of a particular program. Symbolic addresses must then be translated into relocatable (or absolute) addresses by the assembler.

Symbol table A table containing entries and binary values for each symbol defined or used within a module. This table generally contains additional information about the way in which the symbol was defined in the module.

Terminal A unit, normally consisting of both a keyboard and printing (or display) mechanism, that is used to enter information into a computer and to accept output from a computer. When it is used as a timesharing terminal, the computer to which it is connected can be very close or many miles away.

Translate To compile or assemble a source program into a machine language program, usually in the form of a (relocatable) object module.

Trap An unprogrammed conditional jump to a known location, automatically activated by a side effect of executing a processor instruction. The location from which the jump occurred is then recorded. It is distinguished from an interrupt which is caused by an external event.

Trap Servicing Routines Routines that allow programs to handle errors while a program is running. Some of the errors that can be handled in this manner are illegal memory references, and pushdown list overflows.

2's complement A number used to represent the negative of a given value. This number is obtained by substituting a zero for each one and a one for each zero in the bit configuration of the binary number and adding one to the result.

UFD

1. A file whose entries are the names of files existing in a given project-programmer number area within a file structure.
2. The top-level directory for each user. Also, the top-level directory for the ersatz devices which appear as one directory.

Unit The smallest portion of a device that can be positioned independently from all other units. Several examples of units are: a disk, a disk pack, and a drum.

Update To open a file for reading and writing simultaneously on the same software channel, rewrite one or more blocks in place, and close the file.

User's program All of the data and code running in a user virtual address space.

User file directory See UFD.

User I/O mode

1. The central processor mode that allows a user program to be run with automatic protection and relocation in effect, as well as the normal execution of all defined operation codes (including I/O instructions).
2. The monitor mode which allows a job to run with the I/O mode hardware on.

User library Any user file containing one or more relocatable binary modules of which some or all can be loaded in library search mode.

User mode A central processor mode during which instructions are executed normally except for all I/O and HALT instructions, which return control to the monitor. This makes it possible to prevent the user from interfering with other users or with the operation of the monitor. Memory protection and relocation are in effect so that the user can modify only his area of core.

User virtual address space A set of memory addresses within the range of 0 to 256K–1 words. These addresses are mapped into physical core addresses by the paging or relocation-protection hardware when a program is executed.

UUO (Unimplemented User Operations) See programmed operators.

Variable Any entity that can assume any of a given set of values. When stored in memory, a variable can occupy part of a memory location, exactly one memory location, or more than one memory location.

Word An ordered set of bits which occupies one storage location and is treated by the computer circuits as a unit. The word length of the DECsystem-10 is 36 bits. This means that it is possible to store 36 bits of information at each memory address and to transfer all 36 bits between memory and the CPU at the same time.

Working set The collection of pages in physical core for an active job.

The Index